网构软件可信性评估与保障技术

司冠南　著

電子工業出版社
Publishing House of Electronics Industry
北京・BEIJING

内 容 简 介

网构软件是近年发展起来的研究领域，涵盖了软件构件、体系结构、软件开发方法等多方面，并为当前流行的云计算、物联网等概念提供了诸多核心技术。由于网构软件工作于开放、动态、难控的互联网环境，且组成实体多由第三方提供，其可信性问题就变得非常重要，如何保证软件整体及各组成实体的可信性成为网构软件研究领域中一个非常具有挑战性的新问题。本书从网构软件的实体模型、系统结构、软件演化、可信性评估等方面对其可信性评估与保障技术进行了阐述，并提出了解决方案。

本书可供计算机科学、可信计算、服务计算以及相关领域科学研究人员和工程应用人员参考，也可供高等院校和科研院（所）相关专业的教师和研究生阅读参考。

图书在版编目（CIP）数据

网构软件可信性评估与保障技术/司冠南著. —北京：电子工业出版社，2015.1
ISBN 978-7-121-25134-4
Ⅰ. ①网… Ⅱ. ①司… Ⅲ. ①软件设计－研究 Ⅳ. ①TP311.5
中国版本图书馆 CIP 数据核字（2014）第 294368 号

责任编辑：赵 娜　　特约编辑：王 纲
印　　刷：北京季蜂印刷有限公司
装　　订：北京季蜂印刷有限公司
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编 100036
开　　本：720×1000　1/16　印张：13.75　字数：264 千字
版　　次：2015 年 1 月第 1 版
印　　次：2015 年 1 月第 1 次印刷
定　　价：42.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

存在于互联网各个节点上的主体化软件服务，通过多种机制进行协同、整合而形成的软件形态通称为网构软件。网构软件为有效进行异构资源整合、充分利用互联网上大量的软件服务提供了有效手段，但随着对网构软件功能需求的不断增加，系统的结构、体系变得日趋复杂，同时由于软件的运行环境从传统的“封闭、静态、可控”环境转变为“开放、动态、难控”的互联网环境，对可信性保证的要求变得日益突出。目前网构软件可信性保证技术在信任关系的约束机制、推荐信息的准确性、信任衰减参数的合理性、信任演化模型的系统化、可信性评估方法的切实性等方面还存在着不足。针对上述不足，本书在网构软件可信性保证关键技术方面进行了研究，具体研究内容与创新点包括 5 个方面。

（1）在实体个体层面，研究网构软件的可信实体模型。提出了具有自省性、自明性、自主性特点的网构软件强可信智能实体模型。定义了实体可信情况形式化描述语言（EDSADL），使实体能够通过自省机制实时监控与保障自身的可信性水平，并向外界公布以供验证。设计了业务功能模块与可信性保障模块分离的系统结构，保证了实体在向外界公开可信保障机制的同时对自身业务细节的保密性。设计了“环境感知—策略调度—行为触发”的机制，使实体具有感知环境刺激而自主演化的能力，保证了实体在外界环境可信性发生变化时能够做出适时的响应。

（2）在实体间协同层面，研究网构软件实体间的信任约束机制。引入

契约式设计思想，从服务使用者和提供者的角度出发，采用“承诺—评估”机制明确双方的权利和义务，为实体交互过程中的双边规范定义了从低层接口语义到高层可信性情况的约束。定义了信任契约中前置条件、后置条件和不变式三要素的描述方法，保证了服务使用者、提供者以及服务交互关系的可信性，为实体协同、合作和竞争提供了有力的评估依据。

（3）对于信任传递参数，研究实体间信任传递衰减参数的计算方法。综合主观信任与客观评估的优点，提出了基于评估的信任衰减过程。在对实体环境进行可信性评估的基础上，通过逐级计算、信息合并来综合多个推荐者实体的推荐信息，使用对实体间传输可信性的评估结果来修正推荐信息，计算得到信任传递过程中的衰减参数。该方法充分考虑了外部环境、实体自身条件等情况对主观信任值的影响，使信任传递过程中信任衰减参数的计算更加客观、准确。

（4）在系统层面，研究网构软件可信性演化模型。提出了一种对网构软件体系结构进行形式化的建模方法，建立了系统结构模型，并据此提出了基于分层 Petri 网的网构软件可信性演化模型；通过上层 Petri 网对实体之间的各种基本协同关系进行了建模，以此形成系统整体组成结构的描述，反映了实体间信任关系的演化；通过下层 Petri 网对实体内部的契约协商策略进行了建模，设计了基于信任等级带有路径引导的信任契约协商方法。从系统整体的角度建立起动静结合、层次分明、描述统一的网构软件可信性演化模型。

（5）在系统测试与可信性评估层面，研究网构软件系统可信性评估模型。提出了符合网构软件异构性、结构化、动态化特征的可信性评估方法。定义了基于贝叶斯网络的网构软件可信性评估体系，通过树形结构整合了传统的静态指标以及适用于网构软件的动态指标。提出了基于结构模式的可信性评

估指标及其计算方法，在描述各实体之间结构关系、语义关系的基础上，涵盖了对网构软件系统整体及其组成实体可信性的评估。建立起不但能对系统进行评估，还能为用户选择最优化实体提供帮助的可信性评估模型。

本书内容分为 8 章。第 1 章阐述了网构软件可信性相关概念。第 2 章介绍了网构软件可信性保证的相关研究现状，以及本书的主要研究内容、意义和创新之处。第 3 章对网构软件可信实体模型进行了研究，对现有的网构软件实体模型进行了分析，并给出了网构软件强可信智能实体模型及其结构和行为设计。第 4 章对网构软件实体间信任约束机制进行了研究，分析了目前常用的网构软件的信任度量及演化模型，建立了基于契约的网构软件实体交互模型，并提出了基于评估的信任衰减过程。第 5 章是对网构软件可信性演化模型的研究，建立了系统结构模型，根据此模型建立了基于分层 Petri 网的网构软件可信性演化模型，并描述了网构软件系统运行中的可信性演化机制。第 6 章是对网构软件可信性评估模型的研究，建立了基于贝叶斯网络的可信性评估体系，并介绍了评估体系中各指标及系统整体可信性的计算方法。第 7 章是对网构软件动态可信性指标评估技术的介绍，从 Web 应用安全漏洞渗透测试中 SQL 注入攻击建模和 SQL 注入渗透测试用例形式化建模等方面进行了阐述。第 8 章总结了本书已经完成的工作，并对未来的研究做出了展望。

本书第 7 章的编写工作得到了天津工业大学田伟老师的大力支持，在此表示衷心感谢。由于作者水平有限，书中错误与不妥之处在所难免，欢迎各界专家和读者朋友批评指正。

山东交通学院

司冠南

2014 年 10 月

目　录

第 1 章　网构软件可信性相关概念 …… 1

1.1　网构软件 …… 1

1.2　可信计算 …… 8

1.3　软件可信性 …… 11

第 2 章　网构软件可信性研究现状 …… 15

2.1　网构软件可信性保障 …… 15

2.2　网构软件可信性评估 …… 20

2.3　本书的研究内容及意义 …… 22

第 3 章　网构软件可信实体模型 …… 28

3.1　网构软件实体模型 …… 29

3.2　强可信智能实体模型 …… 35

3.3　实体可信情况形式化描述语言 …… 45

3.4　基于 EDSADL 的实体自省机制 …… 57

第 4 章　网构软件实体间信任约束机制 …… 60

4.1　网构软件的信任度量及演化模型 …… 61

4.2　基于契约的网构软件实体间信任关系约束机制设计 …… 63

4.3 基于评估的信任衰减过程 …… 70
4.4 基于网构软件的软件评测支撑平台设计及实验分析 …… 73

第 5 章 基于分层 Petri 网的网构软件可信性演化模型 …… 86
5.1 网构软件结构分析 …… 87
5.2 Petri 网用于网构软件可信性演化的相关研究 …… 89
5.3 网构软件系统建模 …… 96
5.4 网构软件系统运行中可信性演化机制 …… 104

第 6 章 网构软件可信性评估模型 …… 112
6.1 基于贝叶斯网络的网构软件可信性评估体系 …… 113
6.2 可信性评估指标计算方法 …… 122
6.3 网构软件可信性评估实例及结果分析 …… 129

第 7 章 网构软件动态可信性指标评估技术 …… 138
7.1 攻击模型驱动的 SQL 注入渗透测试框架 …… 139
7.2 基于安全目的模型的 SQL 注入攻击建模 …… 147
7.3 SQL 注入渗透测试用例的形式化建模 …… 162

第 8 章 结语 …… 180
参考文献 …… 184

第 1 章

网构软件可信性相关概念

互联网发展日新月异，随着网络上信息量的急剧膨胀，越来越多的用户对网络服务和应用提出了实时、动态、开放、可信的要求。如何整合分布式异构资源，为网络使用者提供快速、高效、可信的动态服务是信息技术发展面临的严峻挑战，也是国家及各行业领域一系列重大需求的共同基础。为了适应这些应用领域及信息技术方面的重大需求，在面向对象、软件构件等技术支持下，形成了基于互联网环境的新的软件形态，即网构软件（Internetware）[1]。由于网构软件工作于开放、动态、难控的互联网环境，且组成实体多由第三方提供，软件的可信性问题就变得非常重要。因此，如何保证网构软件整体及各组成实体的可信性成为网构软件研究领域一个非常具有挑战性的新问题。

1.1 网构软件

为了适应人们不断增加的功能需求，软件系统的结构、体系变得日趋复杂。同时，在飞速发展的互联网技术的推动下，软件的组成逐步由

固定转向多样，软件的行为逐步由静态转向动态，软件的开发逐步由封闭转向开放。从软件实体的角度看，主要经历了对象、构件、服务，并发展了软件 Agent 技术，软件实体的主体化（内容的自包含性、结构的独立性与实体的适应性）程度不断提高；从软件开发的角度看，主要经历了 3 个发展层次：面向对象、基于构件和面向服务[2~11]。

面向对象的方法与技术很好地发展了信息隐蔽原理及抽象数据类型概念，使人们能够从认知的角度控制大型软件设计与开发过程的复杂性。其基于继承的代码复用和多态执行的机制，将各种相关的对象加以有机关联，使所设计的软件对软件工程实践中常遇到的功能易变性具有一定的适应能力。软件的设计与开发也由对功能固定、结构封闭的问题进行自顶向下、逐步精化的结构化方法求解，转变为对功能可变、结构开放的问题进行结构化适应。所使用的开发方法面向具体问题，支撑个体软件开发者，系统开发呈现一阶段集中式结构。

软件构件的概念诞生于 20 世纪 60 年代后期，人们从不同的角度给出了不同的定义，OMG 组织[12]、Carnegie Mellon 大学软件工程研究所（SEI）的 Bachman[13]，以及著名构件学者 Szyperski[14]等均对构件给出了定义。我们认为，Szyperski[14]的定义较好地体现了构件的特征，其定义如下：软件构件是一种用于组装的单元，它具有规范的接口规约和显式的上下文依赖，软件构件可以被独立部署并由第三方组装软件。构件一般由为组装提供基础支撑的构件接口与包含功能具体实现的构件体组成。构件通常比对象具有更大的粒度和更强的复用性，可以在基本不改变内部实现细节的情况下被重复应用于多个软件系统。这样构件就可以由第三方通过一定的外部手段进行一定的剪裁与调节，进而应用于更加广泛与开放的环境中。由此可见，构件能够在解决已有问题的基础上

对未来的同类问题提供支持，满足了大规模工业化生产软件的需求。基于构件软件的开发面向群体问题，支撑群体开发者工业化开发，构件开发与系统组装呈现两阶段分离式结构。

随着互联网的发展，软件的使用环境也逐渐变得更加开放、动态和多变。而开放网络环境中各类资源的共享和集成也对计算机软件技术提出了新的挑战。软件服务的概念就是在这种环境下产生的，并较对象、构件有了新的发展，主要是应对开放网络环境中各类资源的共享和集成对计算机软件技术提出的挑战。首先，软件服务的业务功能相对独立，通常基于可共享或集成的资源构建，因而具有更大的粒度与更强的独立性，更便于使用者直接使用。其次，软件服务具有较高的抽象级别，屏蔽了开放环境带来的异构性问题，且具有松耦合性，可以灵活、动态地组合以生成增值服务。最后，由于开放环境下存在大量类似的资源，使用者可以在量大面广、推陈出新的多个相同或相似服务中不断选择最合适的服务以构建最优的系统。基于服务软件的开发面向问题解，支撑大量软件最终使用者及开发者，服务发布/查询/使用呈现三阶段松耦合结构。

软件 Agent 是近年兴起的一种新技术，通过将人工智能技术与软件技术相结合，建立软件实体内部机制，以使其能够应对开放、动态与难控的外部环境。对于智能 Agent 的认识，Wooldridge 认为[15]“Agent 是这样的一个计算机系统，其位于某个环境中，且能在该环境中采取一定的自治动作以满足其设计目标”。智能 Agent 具有反应性、目标驱动性、社会性，可以感知环境并及时做出反应，以目标为导向驱动行为，相互间进行交互，以此来满足其设计目标[15,16]。这样软件实体就具有了以下能力与特性：对外部环境的感知能力，并通过自身行为对外界产生作用；

推理与决策能力，通过控制与推理使自身行为更具自主性；与其他实体进行有效交互的反应性、目标驱动性和社会性等特性。

在上述技术与方法的支持下，逐渐形成了网构软件的概念。网构软件是由分布在广域范围内、通常由第三方提供、具有主体化服务特征的一组软件实体，通过各类风格各异的协同连接子加以互连互通而形成的联盟，是开放、动态和难控网络环境下的分布式软件系统的一种抽象[17]。它既是传统软件结构的自然延伸，又具有区别于在集中封闭环境下发展起来的传统软件形态的独有的基本特征[18]。

（1）自主性：指网构软件系统中的软件实体具有相对独立性、主动性和自适应性。

（2）协同性：指网构软件系统中软件实体与软件实体之间可按多种静态连接和动态合作方式在开放的网络环境下加以互连互通、协作和联盟。

（3）反应性：指网构软件具有感知外部运行和使用环境，并对系统演化提供有用信息的能力。

（4）演化性：指网构软件结构可根据应用需求和网络环境变化而发生动态演化，主要表现在其实体元素数目的可变性，结构关系的可调节性和结构形态的动态可配置性。

（5）多态性：指网构软件系统的效果体现出相容的多目标性，它可根据某些基本协同原则，在动态变化的网络环境下，满足多种相容的目标形态。

网构软件的组成结构如图 1.1 所示。由图 1.1 可以看出，网构软件是由软件实体层、网构软件层和介于二者之间的标准接口层组成的，它们连同软件用户都是互联网平台的一员。各实体均为散布在互联网的软

件系统，彼此独立、自主运行，不受任何机构或组织的统一控制，在开放、动态和多变的环境中进行协同。

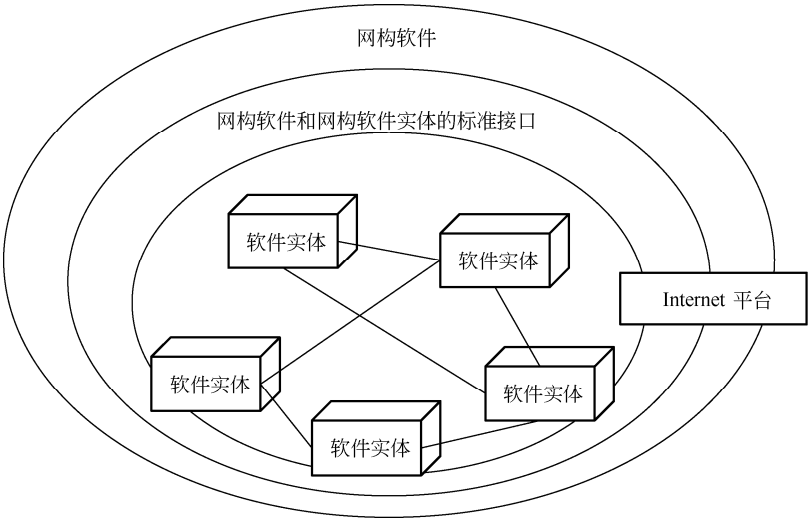


图 1.1　网构软件组成结构图

网构软件的各个实体首先是一个构件，具有与传统软件构件类似的业务接口及构件体内部实现。这样各实体可以独立部署、运行与演化，并通过业务接口提供计算、控制或连接等功能。但互联网“天然”的异构性决定了网构软件实体并不等同于传统软件构件，其具备个体感知能力、环境感知能力和个体适应能力等。这些能力主要以反射接口所提供的对网构软件实体自身业务的监测、分析、规划、调整等反射功能为基础，进而实现网构软件整体的自主性、演化性、协同性、多态性和反应性等特性。对于网构软件整体而言，既可以作为一个部署于互联网上的软件系统为处于不同时区的用户服务，也可以作为一个软件实体对随时到达的其他网构软件的请求进行响应。从长期来看，网构软件的动态模型使其能够自主演化，主动适应开放、动态和多变的互联网环境，以满

足某一时段内用户的个性化需求。并且根据环境的变化随时对运行系统按照所需的指标加以调整与演化，从而使系统在下一个时段能更好地满足用户的需求。

网构软件的研究工作借鉴和涵盖了当前最新的研究理念与成果，主要包括基本理念、实现主体、软件协同、运行机制、安全保障、系统管理自主化等多个方面。

在基本理念方面，英国学者提出的 GUC（Global Ubiquitous Computer）理念[19]从概念的角度，论述了包含 UC（Ubiquitous Computer）和 Global Computing 的观念；Shaw[20]等从资源集成与共享的角度提出了 ORC（Open Resources Coalitions）的概念，ORC 是指分布自主的资源所动态形成的、面向特定任务的临时联盟，这些资源可以是信息、计算、通信、控制或服务；浙江大学的张三元教授及其研究团队建立了在 P2P 网络拓扑结构上的网构软件模型 P2P-IWRM，通过引入构件副本、构件复用形式及构件副本测试状态，建立了本地构件库和网络构件库，增强了网构软件的自适应性[21]。

在实现主体方面，北京大学的梅宏教授[22,23]及其研究团队通过改装已有普通构件，将 Agent 技术与构件技术结合起来，为构件定制行为规则和规划，使之具有自主性，并提出了一种基于自主构件的协作框架，借助环境改变来引导自主构件间的协作；吕建等[24]以开放网络环境下的网构软件需求为切入点，基于软件 Agent 的原理、方法和技术，提出了一种开放协同软件模型来作为网构软件的基础模型，并在此基础上提出了一条建立基于 Agent 的网构软件模型的技术途径，即网构软件模型=开放协同模型+环境驱动模型+智能可信模型；常志明等[25]基于动态 Agent 理论为网构软件建模，描述了构件的运行状态、自主运行及自适应演化

运行机制；王怀民等[26]提出了一种支持软件环境适应能力细粒度在线调整的构件模型 ACOE，将软件环境适应能力中的感知、决策、执行等关注点封装为独立的构件和连接子，通过动态软件体系结构技术来支持它们的在线重配置。

在软件协同方面，Carriero 和 Gelernter 提出了 Programming = Coordination + Computation 的理念，即分布式程序设计中将协同与计算分离[27]，在此理念指导下，Linda 语言使用基于共享空间的协同语言或协同框架，这类协同模式解除了各个软件实体之间的直接耦合，而使用共享空间来进行基于通信内容的匿名通信；金芝教授[28]及其研究团队为构件建立环境本体，以环境为桥梁实现需求和构件的相互理解，以环境的状态变迁为驱动进行构件的组合。

在运行机制方面，构造能够热演化或自演化的软件系统难度非常大，SA（Software Architecture）[29]技术的发展与成熟为研究软件系统的自演化提供了结构化的基础。演化方面的技术较多地采用框架结构方法，通过对框架结构的改变来演化系统，也有一些做法是采用动态转化（Software Dynamic Translator）[30]等。南京大学吕建等[31,32]对软件热演化进行了研究，开发出了自演化软件支撑平台 Artemis-MAC。

在安全保障方面，Blaze[33]在 1996 年首次明确提出可信管理，主要使用早期的分布式身份认证技术，并在单纯的身份认证的基础上进行了扩充。吕建等[34]针对网构软件的特性，设计了一个基于层次式 Monitor 的信任管理系统。可信管理与可信评估的有机结合将是研究的重要方向，Grandison 等[35]提出了一个更为一般的可信管理的定义将可信管理与可信评估统一起来。

在系统管理方面，提出了自治式计算的概念。其中，文献[36]介绍

了一种基于 Agent 的软件框架结构，软件 Agent 被用以封装系统内的各个组件，系统的自主管理通过 Agent 的自主性得以实现。Hot Swapping 技术[37]利用代码插入（Code inter-positioning）和代码替换（Code Replacement）等技术实现了构件的自监控等自管理机制。

1.2 可信计算

由于互联网环境的开放性、动态性、多变性，可信问题越来越受到人们的广泛关注。可信的概念来源于可信计算，并在诸多领域得到了大力推广。

可信计算是一种包括可信硬件、可信软件、可信网络和可信计算应用等诸多方面的新的信息系统安全技术，其思想源于人类社会，把人类社会成功的管理经验用于计算机信息系统和网络空间，以确保计算机信息系统和网络空间的安全可信[38]。可信计算组织（TCG）认为[39]，可信计算的总体目标是提高计算机系统的安全性，现阶段的主要目标是确保系统数据的完整性、数据的安全存储和平台可信性的远程证明。

可信计算的基本思想如下[39]：

（1）首先在计算机系统中建立一个信任根，信任根的可信性由物理安全、技术安全与管理安全共同确保；

（2）再建立一条信任链，从信任根开始到硬件平台、操作系统，再到应用，一级测量一级，一级信任一级，把这种信任扩展到整个计算机系统，从而确保整个计算机系统的可信。

TCG 的信任链如图 1.2 所示。

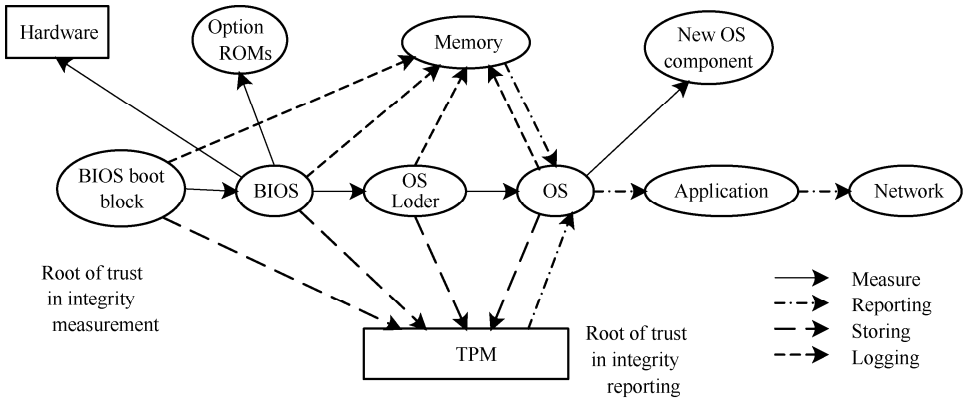


图 1.2　TCG 的信任链

自 20 世纪 70 年代初期 Anderson 首次提出可信系统（Trusted System）的概念以来[40]，可信计算就一直受到学术界和工业界的广泛关注，国内外研究者进行了大量研究工作。1983 年美国国防部制定了世界上第一个《可信计算机系统评价准则》[41]，其中第一次提出了可信计算机和可信计算基（TCB）的概念，并把 TCB 作为系统安全的基础。之后，美国国防部又相继推出了可信网络解释（TNI）[42]和可信数据库解释（TDI）[43]，从而形成了最早的一套可信计算技术文件。1999 年，IBM、HP、Intel 和微软等著名 IT 企业发起成立了可信计算平台联盟（TCPA），2003 年 TCPA 改组为可信计算组织（TCG），标志着可信计算技术和应用领域的进一步扩大，形成了可信计算的新高潮。TCG 已经制定了一系列的可信计算技术规范[44,45]，如可信 PC、可信平台模块（TPM）、可信软件栈（TSS）、可信网络连接（TNC）、可信手机模块等，而且不断地对这些技术规范进行修改完善和版本升级。欧盟于 2006 年 1 月启动了名为“开放式可信计算”的可信计算研究计划[46]，有几十个科研机构和工业组织参加研究。目前，可信计算已经成为许多国际学术会议的重要议题。

在国内，2003 年武汉瑞达公司和武汉大学合作研制出我国第一款可信计算平台模块（TPM）和可信计算平台[47]。2006 年我国进入制定可信计算规范和标准的阶段，在国家密码管理局的主持下制定了《可信计算平台密码技术方案》和《可信计算密码支撑平台功能与接口规范》[48]。2007 年在国家信息安全标准委员会的主持下，我国开始制定一系列的可信计算标准，包括芯片、主板、软件、可信网络连接等标准，国家自然科学基金委启动了“可信软件重大研究计划”[49]。 2008 年中国可信计算联盟（CTCU）成立，我国的可信计算事业进入了蓬勃发展的阶段。我国的可信计算技术与产品得到了国际同行的高度评价，我国已经位于国际可信计算领域的前列。

可信计算的研究主要包括可信计算理论、可信平台模块、可信计算平台、可信计算平台测评、可信软件、可信网络连接与远程证明等方面。可信计算理论[39]主要研究可信的定义与属性、信任度量模型、可信度量理论与技术等方面，为可信计算的研究与发展提供理论基础；可信平台模块（TPM）[44,45]是可信计算平台的信任根，是可信计算的关键技术之一；可信计算平台[50]包括可信 PC、可信 PDA、可信服务器等，是对可信计算技术的具体化应用；可信计算平台测评[51]不仅为用户选择可信计算机产品提供了依据，也为可信计算平台技术及产品的改进和发展提供了依据；可信软件[52]的研究是从软件可信性的建模、设计、验证、演化入手，以解决传统的“面向正确性的软件理论+工程化”模式难以适应开放环境下软件系统开发的问题；可信网络连接与远程证明[53,54]是把可信扩展到网络，使网络成为一个可信的计算环境，以应对各种安全风险与威胁。

1.3　软件可信性

随着软件规模的不断扩大，其内部结构越来越复杂，应用环境越来越开放，软件在系统中的作用越来越重要，其复杂性不断增强，软件可信性问题也日益突出，这些因素使得人们更加关注软件的可信性问题[55]。自 20 世纪 70 年代初期可信系统的概念产生以来，人们从不同的角度对可信的概念提出了多种表述：从系统的角度，ISO/IEC 15408 标准[56]将可信定义为一个可信（Trusted）的组件、操作或过程的行为在任意操作条件下是可预测的，并能很好地抵抗应用软件、病毒以及一定的物理干扰造成的破坏；可信计算组织 TCG 认为[57]，如果一个实体总是按照其设定目标所期望的方式行事，则称这个实体为可信的；从用户体验的角度，微软公司的比尔·盖茨[58]认为可信计算是一种可以随时获得的可靠安全的计算，并包括人类信任计算机的程度，就像使用电力系统、电话那样自由、安全；从网络行为的角度，林闯等[59]认为可信的网络应该是网络系统的行为及其结果是可以预期的，能够做到行为状态可监测、行为结果可评估、异常行为可控制。

对软件系统来说[49]，其可信性通常是指在特定环境下软件运行行为及其结果符合人们预期，并在受到干扰时仍能提供连续服务的性质。传统意义上的软件可信性（Trustworthy）主要包括安全性（Security）和可依赖性（Dependability）两个方面[60]，如图 1.3 所示。

可依赖性指软件提供服务的能力能够被证实是可以信任的，是一个包含如下方面属性的集成概念。

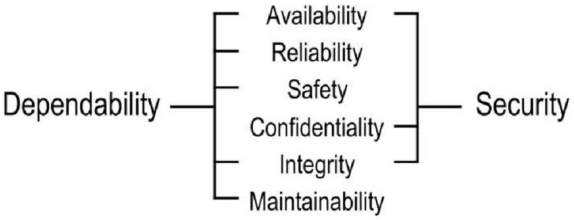


图 1.3　可信性属性

- 可用性（Availability）：可随时提供正确服务。
- 可靠性（Reliability）：保证正确服务的连续性。
- 防危性（Safety）：对用户和环境不会造成灾难性后果。
- 完整性（Integrity）：没有不当的系统变更。
- 可维护性（Maintainability）：可进行修改与维修的能力。

安全性被定义为由保密性、完整性、可用性同时作用而成的复合概念。其中，保密性（Confidentiality）定义为未经授权的信息不被泄露；完整性和可用性则被赋予了安全环境下的定义。

- 完整性（Integrity）：没有未经授权的系统变更。
- 可用性（Availability）：只对已授权的行为随时提供正确服务。

软件可信性与软件行为、应用需求有紧密联系，随着信息技术、互联网等各类软件应用技术的发展，以及软件形态和作用的变化，软件可信性也被赋予了不断丰富与发展的内涵。

经过近 40 年的发展，互联网已经成为人们工作、生活中不可或缺的重要部分。互联网本身的开放性、动态性及其资源的成长性、自治性、多样性，给基于互联网的软件带来了更大的不可控性和不确定性；同时，随着电子商务、电子政务、跨域的资源共享等新的应用模式不断涌现，应用规模的不断扩展，所涉及资源的种类和范围的不断扩大，

应用复杂度的不断提高，以及计算模式的不断革新，互联网环境下软件系统的可信性已经成为一个亟待解决的问题[61]。为此，人们逐渐将社会系统中的可信性概念融入传统的软件可信性概念中，从资源共享、服务提供和服务访问三个角度做出了新的诠释。从资源共享的角度，软件可信性要求其对于资源的访问和利用总是处于规则允许的范围之内，并通过对参与实体的授权和身份验证来确保系统的正常运行，也即身份可信；从服务提供的角度，软件可信性要求软件系统的功能是可依赖的，其服务质量能达到其承诺的水平，也即能力可信；从服务访问角度，软件可信性要求参与实体要遵守一定的行为规则，既包含对实体特定行为的约束，也包含对实体间协同过程中所应遵循的信息交互规范，也即行为可信[62]。

由此可见，目前的软件可信性概念已经包含了内部属性和外部表现两方面的含义。

从内部属性视角来看，软件可信性是对一个软件系统的多个自身属性的综合反映，包括可靠性、可用性、完整性等。软件系统通过这些属性来证明自身可以满足用户的需求，而用户则可以在使用软件系统前通过这些属性来评估系统行为是否能够符合自己的预期。这称为对软件系统可信性的事前评估。

从外部表现视角来看，作为更大范围内协作活动的一个组成实体，由于软件系统内部机制的不可见性，其可信性更多地表现为协作过程中的行为是否符合用户的预期。也就是说，通过协作，用户（人或者其他实体）可以对软件的表现得出一个评价结果（或称信任值）。这称为对软件系统可信性的事后评价。

软件可信性的研究主要包括：软件可信性建模，包括对期望系统行

为的描述模型[63]、系统可信性的度量模型[64]等，这是软件可信性的基础；软件可信性设计，包括软件体系结构设计与演化[65,66]、可信性动态保证[67]和软件容错技术[68]等，这是软件可信性的保证；软件可信性验证与评估，包括软件分析[69]和模型检验[70]等，这是对软件可信性保证成果的检验；软件可信性演化，包括社会软件工程[71]、软件自适应演化[72]、软件可信评估与选择[73]、超大规模系统开发与演化[74]等，这是软件可信性的持续保证。

第 2 章

网构软件可信性研究现状

网构软件的概念是由中科院院士杨芙清教授[18]首先提出的，引起了越来越多的研究机构和学者的关注。目前，国内外对于网构软件可信性的研究仍处于起步阶段。对应于软件可信性的两方面含义，网构软件可信性保证的研究也分为基于网构软件外部表现的可信性保障与基于网构软件内部属性的可信性评估。

2.1 网构软件可信性保障

网构软件是由存在于互联网中异构、自治的软件实体组成的，传统的软件可信性保障技术难以适用于网构软件平台，主要表现在[75]：①网构软件主要是由大量相互协同的软件实体组合而成的，在构建过程中涉及大量的移动实体与匿名实体，开发者难以直接对其进行管理；②网构软件在运行时具有结构上的动态演化性，用户需求的变更或环境的变化都会使系统对自身进行重新配置，从而增加、删除或替换参与协同的实体；③网构软件的用户具有开放性、多样性的特点，各用户的需求在不同环境、不同时

间也不尽相同，与传统软件相比更具有“柔性”特点；④网构软件的开放性与动态性也导致了不会有绝对可信的第三方来对所有参与协同的软件实体的可信程度做出担保。

因此，网构软件需要一种动态演化的、相对的、“柔性”的可信性保障机制，利用实体间交互行为等外部表现而形成的信任关系来解决上述问题成为目前研究的一个热点。其基本思想是用基于黑盒的柔性可信保障技术作为传统基于白盒的刚性保障方法的补充，如图 2.1 所示。

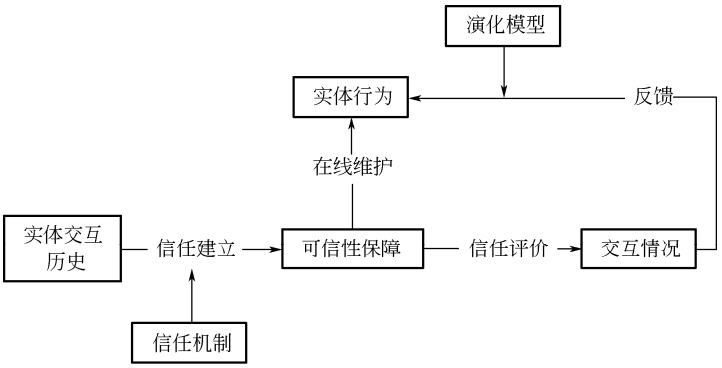


图 2.1　网构软件可信性保障模式

利用观测、推荐等手段获得相关实体交互的历史信息；采用概率统计、信息论等方法对其未来行为的信任度做出推断，结合相应的信任机制建立信任关系；通过评价实际交互情况得到对当前信任关系的反馈，结合预定的信任演化模型在线维护实体行为，并对信任关系进行演化。

对于信任，目前尚缺乏精确的、广泛可适用的定义，在开放协同环境下，信任一般被定义为网构软件中实体关于其他实体或实体集合具有完成某一特定任务能力可能性的主观判断[75]，其程度依赖于实体对于信任对象的直接经验和推荐信息。信任关系通常分为两类：直接信任关系与推荐信任关系。直接信任关系是指实体与协作者之间的信任关系，表明实体认为

协作者成功参与下次协作的概率，通过评估协作者执行某类操作交互的成功次数与失败次数得出；推荐信任关系是指实体与推荐者之间的信任关系，体现了实体认为其所提供信任信息为真的程度，通过对推荐者信任度的计算得出。在这方面所涉及的主要关键技术包括信任管理和信任评估两个侧面。

业内公认最早的对信任管理（Trust Management）的定义是由 Blaze 等人[33]提出的“采用一种统一的方法来描述和解释安全策略（Security Policy）、安全凭证（Security Credential）以及信任关系，用于对关键性安全操作的直接授权”。基于信任管理的基本思想，出现了一批早期的信任管理系统，如 PolicyMaker[33]、KeyNote[76]、REFEREE[77]、RT[78]和 dRBAC[79]，其关注的主要技术在于安全策略及安全凭证的表述和决策过程中的分布式信息收集算法。近期信任管理方面的研究如由欧盟赞助的 SECURE 项目[80]、斯坦福大学 Kamvar 等人提出的 P2P 全局信誉系统[81]，国内南京大学的吕建等人[82]、北京邮电大学信息安全中心[83,84]也对信任在多种开放网络环境下的应用进行了研究。

信任评估的思想起源于互联网中大规模的资源共享与集成的应用需求，由于在资源集成与共享的过程中会碰到大量的匿名实体和移动实体，因此基于身份认证的信任管理技术将很难适应开放协同环境的需求[1]。而且信任是模糊、非理性的，是一种经验的体现，对信任的评估应有内容和程度上的划分[85,86]，因此采用精确的、绝对理性的方式来描述和处理复杂的信任关系具有一定的局限性；借助熟识实体的推荐和自身的经验信息对陌生实体的可信度做出评价，然后依据信任度进行决策是开放协同环境中一种潜在的有效决策机制。由此产生了信任评估的思想，并出现了一些比较著名的信任评估模型，如 Abdul-Rahman 模型[87]、Beth 模型[88]、Jøsang 模型[89]等，这些模型面向解决传统网络的安全问题，以人类社会中的信任

为基础，给出了信任信息数学抽象，并对软件实体的信任进行数学建模，使得软件实体的信任可用机器表示，同时给出了处理信任信息收集与合并的方法。不足之处在于这些模型过多地强调了范性，力图适用于更为广泛的网络环境，而且无法很好地区分诚实实体和恶意实体，致使模型在面临恶意攻击时毫无抵抗能力。在这些模型的基础上，南京大学吕建及其研究组开展了一系列工作，针对软件服务的特性给出了 TEM[75]和 DTME[90]两个信任模型，分别用于处理推荐信任的收集与合并以及形成软件实体联盟，在可操作性和环境适应性方面较之前的研究工作有了一定改善。随着电子商务、P2P 等新的应用与技术的发展，新出现了一批针对动态网络的实体行为可信性预测和管理模型，比较典型的有 eBay 系统中的 TMBS 模型[91]、EigenTrust 模型[92]、PowerTrust 模型[93]、PeerTrust 模型[94]、R^2BTM 模型[95]、Dirichlet 模型[96]等。这些模型考虑了信誉、风险、反馈等多项信任评估中的影响因素，对直接信任与推荐信任进行综合，能够对开放网络环境下实体的信任度情况做出较好的评估。

本质上，可信管理与可信评估有其相同之处，均是利用实体间的信任关系解决开放协同环境中的可信问题。这些模型的基本思想都是基于信任网络中的信任度量及演化模型，该思想对实体间信任关系的建模如图 2.2 所示。此模型形象地刻画了网构软件实体间的信任信息传递过程，有效地抽象出实体间的信任关系，但对于网构软件可信性保障来说仍存在着一些不足，具体表现在以下几个方面。

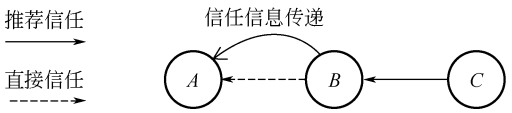


图 2.2　信任信息传递模型

（1）信任信息主要来源于对网构软件实体进行交互的历史结果。由图 2.2 可见，信任度量与演化模型中的信任关系分为直接信任与推荐信任。直接信任指的是实体 *B* 具有对实体 *C* 的某类经验信息，以经验作为信任评估的主要依据。推荐信任是一个实体对另一个实体所推荐的经验信息的可信程度（或采纳程度），即实体 *A* 对实体 *C* 的信任信息是由实体 *B* 将其对实体 *C* 的直接信任推荐给实体 *A* 而获得的，同样是该实体历史经验的反映[87]。此评估过程主要依赖对目标实体的访问历史记录，而很难在与一个全新的目标实体进行实际交互之前测得其可信性。同时，在对一个网构软件系统整体进行初始化安装或部署时，各实体间的信任关系也只能通过默认值进行设置，而不能进行准确的度量。另外，信任推荐的机制主要通过评估者接收推荐者的推荐信息而与目标实体建立信任关系，是一个单方面的过程，且没有显式的机制对此关系进行约束，这对系统整体的可信性保证有着很大的影响。

（2）推荐信息质量的度量依赖于事后评估的方式。信任关系的演化是指随着协作的进行而自动形成信任关系与更新已有的信任关系。图 2.2 中，在每次协同之后，评估者 *A* 先根据交互结果更新自身对评估对象 *C* 的直接信任，然后根据直接信任评价推荐信息，并依据对推荐信息的评价来更新推荐者 *B* 的信任度。在这种方式下，评估者首先需要与推荐者推荐的实体进行实质性的业务交互，并根据实际的交互结果得出最终的信任值，然后才能对推荐信息的质量做出度量。如果评估者受到恶意推荐者的误导而选择了错误的实体，则在完成业务交互后才有可能发现问题，这也会对系统业务数据的安全与系统整体可信性的保障造成很大影响。

（3）信任传递参数对实体个体所处环境差异考虑不足。当多个实体建立起信任信息传递的信任链时，推荐信息信任度具有随着信任链增长而衰

减的特点。这种衰减参数的设置，要么采用设置默认值方法，要么通过评估者对推荐者的信任度计算。而对于网构软件，各实体间信任信息的衰减除受信任传递路径长度的影响外，还会受到评估者到目标实体间与推荐者到目标实体间通信链路、信息传递可信性等差异的影响。由于评估者、推荐者和评估对象所处的网络位置不同，各实体之间的通信链路等环境也会有所差异。而这些差异对系统整体可信性的影响也是显而易见的。因此，信任传递参数中还需要加入对各实体间通信环境情况的评估数据。

2.2 网构软件可信性评估

由于网构软件是近些年提出并发展起来的，专门针对其内部属性进行可信性评估的研究成果不多，多是基于构件、Web 服务可信性评估的思想。

网构软件实体本身也是一个构件，因此可沿用构件可信性评估与度量方法，2003 年 Meyer 提出的可信构件的 ABCDE 模型[97]是较为权威的可信构件框架模型，但未给出可信构件的度量方法。ISO/IEC9126[98]的软件质量模型的贡献在于将软件质量特性分为外部特性和内部特性，考虑了软件产品不同生命周期阶段的不同形态问题。Yacoub 等[99]用构件依赖图来表述构件间的组装交互关系，提出了一种针对层次软件的可靠性模型。Hamlet 等[100]给出了可信构件的度量模型，为可信构件的研究提供了定性的参考。Roshandel 等[101]提出了基于可信构件软件系统的四视图模型 Quartet，为可信构件的度量模式指明了发展方向。Reussner 等[102]关注构件在对外交互接口上所表现出的可靠性，给出了评估构件在不同应用域中

可靠性值的方法，但并未对构件间协同连接的可靠性情形进行分析。Guerra[103]等提出了理想容错构件 IFTC，给出了构件系统的异常处理方式。Bobbio[104]等通过对比贝叶斯网络和故障树在可信性评估方面的优劣，证明了贝叶斯网络在构件系统可信性研究上的可行性。在国内，吴国全等[105]提出了一种基于贝叶斯网络的质量评估模型，以及在此基础上的构件服务质量动态评估方法。郭树行等[106]通过分析可信构件研究的若干领域，总结出构件可信性的 3 个角度。毛晓光等[107]提出了基于构件软件的一个可靠性通用模型，关注动态开发过程中的可靠性跟踪。陈锦富等[108,109]基于错误注入的方式，研究构件安全测试错误注入模型，并给出了错误注入测试用例生成算法。

对服务可信性评估的研究主要关注服务的质量（QoS）[110]，认为服务质量的高低直接影响服务的可信性，即服务质量参数高、质量好的服务更为可信[111,112]。其方法主要是基于监测服务的 QoS 属性来度量服务的可信性[113~115]。另一方面，结合 Agent 的研究，Maximilien 等人[116]用服务质量作为信任评估的依据，将实际的服务在其质量各个维度上达到服务提供方所宣称的程度，作为服务可信性的程度，提出用 Agent 帮助服务提供方与服务请求方进行动态服务选择。在国内，金芝等[117]基于一个 Agent 的信任本体，提出了一系列基于信任推理的计算规则支持信任值的计算，帮助服务 Agent 进行理性的选择决策。刘旭东等[118]在基于可信证据的 Web 服务可信评估机制的基础上，提出了一种基于用户反馈的上下文敏感可信评估方法，进一步提高了 Web 服务可信评估的准确性。王怀民等[61,62]提出了 iVCE 评价体系，将实体的身份信任、服务的 QoS 以及实体间的信任推荐综合起来。

通过研究国内外发展现状可以发现，已有研究成果主要针对传统软件

系统特性，关注面向封闭或单一管理域的软件安全、可靠和可用等问题。而网构软件具有节点高度自治性，节点链接开放性和动态性，人、设备和软件多重异构性，实体行为不可预测性，运行环境潜在不安全性，使用方式个性化和灵活性，网络连接环境复杂性，使用资源的成长性、自治性和多样性等特征。传统的软件系统可信性评估方法已不能很好地适应网构软件可信性评估的要求，急需从安全机制的灵活性以及面向开放环境的软件能力评估方面进行扩展。

2.3 本书的研究内容及意义

2.3.1 研究内容

通过研究国内外发展现状与趋势，可以发现，网构软件的研究虽然在基本理念、软件实体、软件协同、运行机制、安全保障、系统管理自主化、技术体系系统化等多方面都取得了一定成果，但在可信性保证方面仍存在不足：①实体间信任关系的建立仅仅依靠其他实体的主观推荐信息，缺少对于客观评估的有效利用；②信任的源头依赖于对其他实体的访问历史记录，而很难在实际交互之前测得推荐信息的准确性；③信任的传递参数只是基于推荐信任的自然衰减，对不同实体的具体位置、网络环境等实际情况考虑不足；④在信任的协商与演化过程中，缺少一个系统级模型对实体内部与实体之间运行机制的变化情况进行统一描述与持续管理；⑤传统的可信性计算通常站在某个软件实体的角度，通过考察系统的输入和输出参数、运行状态、接口信息等方面因素，计算其他实体的可信度，而没有站

在第三方角度深入分析系统结构，且主要针对传统软件系统特性。

因此本书基于已有的网构软件实体模型及信任度量与演化模型，引入可信计算中建立信任链模型的思想，建立具有自省性、自明性、自主性的强可信智能实体模型；参照信任链模型建立基于信任契约的实体间信任关系模型及基于评估的信任衰减过程，力图将主观评价与客观评估结合起来；构建网构软件系统实体内部机制与实体间交互关系演化的统一描述模型；针对传统可信性评估方法在对网构软件进行评估时的不足，提出适应网构软件特征的可信性评估模型。具体研究内容包括：

（1）在实体个体层面，研究网构软件的可信实体模型。该模型应能够实时监控与保障实体自身的可信性水平，首先确保自身行为的可信；拥有对实体自身功能及可信性情况的描述机制，能够向外界提供，并且可以方便、安全地进行验证；可信性保障功能具有感知环境刺激并进行自主演化的能力，保证实体在外界环境可信性发生变化时能够做出适时的响应。通过上述研究建立起具有明确可信保障的实体模型，为前述第①个问题提供解决途径。

（2）在实体间协同层面，研究网构软件实体间的信任约束机制，该机制作为实体交互过程中的双边规范，不仅需要包括低层接口语义定义，还需要对高层可信性情况进行约束，以便为实体协同、合作和竞争提供有力的评估依据。研究实体间信任传递衰减参数的计算方法，该方法应充分考虑外部环境、实体自身条件情况对主观信任值的影响，以使信任传递过程中信任衰减参数的计算更加客观、准确。通过上述研究建立起具有显式约束的可评估的实体间信任机制，以解决前述第②个和第③个问题。

（3）在系统层面，研究网构软件可信性演化模型。分析网构软件的系统结构，研究如何使可信性演化模型符合网构软件的结构与行为特征，涵

盖系统整体组成与各实体内部契约协商策略，并在实体间的信任关系与实体内部机制两方面，对信任的建立与演化过程进行持续管理与统一描述。通过上述研究建立起动静结合、层次分明、描述统一的网构软件可信性演化模型，以解决前述第④个问题。

（4）在系统测试与可信性评估层面，研究网构软件系统可信性评估模型。研究如何使该模型符合网构软件的组成结构、开发方式及演化特征，能够涵盖对网构软件系统整体及其组成实体可信性的评估，兼顾评估指标的静态特征与动态特征。通过上述研究，建立起不但能对系统进行评估，还能为用户选择最优化实体提供帮助的可信性评估模型，以解决前述第⑤个问题。

本书在以上方面进行了深入研究，并在关键技术上取得了重要突破，主要创新点如下：

（1）在实体个体模型层面，提出了具有自省性、自明性、自主性特点的网构软件的强可信智能实体模型。定义了实体可信情况形式化描述语言（EDSADL），使实体能够通过自省机制实时监控与保障自身的可信性水平，并向外界公布以供验证；设计了业务功能模块与可信性保障模块分离的系统结构，保证了实体在向外界公开可信保障机制的同时对自身业务细节的保密性；设计了“环境感知—策略调度—行为触发”的机制，使实体具有感知环境刺激而自主演化的能力，保证了实体在外界环境可信性发生变化时能够做出适时的响应。这种自说明、可评估、能演化的强可信实体，为前述第①个问题提供了解决途径，并建立了信任的基点，即信任根。

（2）在实体间协同层面，提出了基于信任契约的网构软件实体间信任关系约束机制。将契约式设计引入实体间信任关系约束机制中，从服务使用者和提供者的角度出发，采用“承诺—评估”机制明确双方的权利和义

务。定义了信任契约中前置条件、后置条件和不变式三要素的描述方法，保证了服务使用者、提供者以及服务交互关系的可信性。这样就构成了信任链模型的第二个组成部分，即信任链传递过程中的可信认证机制，也解决了前述第②个问题。

（3）对于信任传递参数，综合主观信任与客观评估的优点，提出了基于评估的信任衰减过程。不同于信任链模型自底向上、逐级认证的模式，实体间信任关系存在着多个推荐者，并且信任随着信任链逐级递减。因此，本书在对实体环境进行可信性评估的基础上，通过逐级计算、信息合并来综合多个推荐者实体的推荐信息，使用对实体间传输可信性的评估结果来修正推荐信息，计算得到信任传递过程中的衰减参数，从而解决了前述第③个问题。

（4）在软件系统层面，提出了一种对网构软件体系结构进行形式化的建模方法，建立了系统结构模型[119]，并根据此模型提出了基于分层 Petri 网的网构软件可信性演化模型。该模型在上层 Petri 网中对实体之间的各种基本协同关系进行了建模，以此形成系统整体组成结构的描述，反映了实体间信任关系的演化；在下层 Petri 网中，对实体内部的契约协商策略进行了建模，设计了基于信任等级带有路径引导的信任契约协商方法。通过此模型从系统整体的角度对网构软件实体间的信任协商与演化过程进行了统一描述、持续管理，解决了前述第④个问题。

（5）在网构软件系统测试与可信性评估层面，提出了符合网构软件异构性、结构化、动态化特征的可信性评估方法[120]。定义了基于贝叶斯网络的网构软件可信性评估体系，通过树形结构整合了传统的静态指标以及适用于网构软件的动态指标。提出了基于结构模式的可信性评估指标及计算方法，在描述各实体之间结构关系、语义关系的基础上，涵盖了

对网构软件系统整体及其组成实体可信性的评估。该模型不但能对系统整体进行评估，还能为用户选择最优化实体提供帮助，解决了前述第⑤个问题。

2.3.2 研究的理论与实际意义

网构软件的出现，为异构资源的整合、基于互联网软件服务的充分利用提供了良好平台。由于其具有复杂性、开放性、演化性等特点，加上由网络交互、共享和协同所带来的众多不安全因素的影响，可信性保障问题就变得更加突出。但是网构软件技术发展时间不长，实体模型、交互机制、系统平台、开发方法等还没有形成统一、成熟的标准，可信性保障机制也还不够完善。因此，网构软件可信性保障技术的研究，对于推动软件技术与方法从面向“封闭、静态、可控”环境转向“开放、动态、难控”环境，有着重要的理论价值。

另一方面，由于网构软件是一种由分布在广域范围内、由第三方以服务形式提供的一组软件实体的联盟，其中多数服务的源代码和内部结构通常不可见，只能通过预先定义好的接口规范进行访问。同时为了保证网构软件的个性化和灵活性，其开发、部署、运行和维护的环境从传统的封闭、静态、可控环境转变为开放、动态、难控的互联网环境，其可信性评估就变得非常重要。而传统的软件可信性评估方法无法适应网构软件的自主性、演化性、协同性、多态性等特性。因此，研究新的适用于网构软件的可信性评估方法，对于有效整合、充分利用互联网上的多种异构资源有着重要的应用价值。

在网构软件可信性保障方面，本书从实体个体模型、实体间协同关系、

软件系统三个层面提出了可信性保障模型，为网构软件可信性保障关键技术的研究做出了贡献。在网构软件可信性评估方面，本书从网络这个大环境出发，充分考虑网构软件系统的特点，提出了完整的可信性评估体系，为服务实体及网构软件系统整体的可信性全面评估提供了技术支持。

第 3 章

网构软件可信实体模型

作为网构软件的基本组成部分，软件实体以封装了各类资源的软件服务的形式存在。与传统的软件实体不同，开放网络环境下的软件服务通常由第三方提供，具有相对独立、面向多用户的个体特征。而实体间的协同，也是在分布环境中的松散的服务调用者与被调用者关系。而且，调用者所需的功能可能难以直接由一个被调用者完成，而需要通过若干个被调用者的多个功能组合才能得以实现。因此，网构软件实体模型需要做到既能够保障本实体自身可信性，又能对实体间信任关系及协同行为予以支持，同时具有能够根据外界环境情况实时调整自身行为的演化机制。本章将对网构软件可信实体模型进行研究，提出具有自省性、自明性、自主性的网构软件强可信智能实体模型，进行模型的结构及行为设计，并定义 EDSADL 作为网构软件实体内部可信性的描述机制及实体内部与实体之间可信性的传递途径。强可信智能实体的建立，形成了最初的信任根，为信任链的建立奠定了良好的基础。

3.1　网构软件实体模型

软件结构模型决定了软件的基本形态，是软件方法学的核心，主要包括软件实体和协同方式两部分；对于网构软件来说，软件实体以封装了各类资源的软件服务的形式存在[24]。目前针对网构软件实体模型的研究主要包括：基于 Agent 的网构软件实体/构件模型[24,25]、基于自主构件的网构软件实体模型[22,23]、支持软件自适应的网构软件实体模型[26]。

3.1.1　基于 Agent 的网构软件实体/构件模型

网构软件的实体具有相对独立性、主动性和自适应性，能够感知外部系统运行，并对环境的变化做出响应。而 Agent 又是具有反应性、目标驱动性、社会性，并在其所处环境中采取一定的自治动作以满足其设计目标的系统。因此，基于软件 Agent 的原理、方法和技术来设计网构软件的实体模型便成了最直接、最自然的选择。

文献[24]提出了一种开放协同软件模型作为网构软件的基础模型。其核心是针对开放网络环境的需要，给面向对象程序模型中的核心概念“OM”赋予新的含义，如表 3.1 所示。

为了满足开放网络环境对方法调用多样化的需求，文献[24]提出了基于移动 Agent 的协同程序设计技术：定义两类表格（GROUP 表与 LOCATION 表），将开放环境下的方法调用进行分解，并按照顺序

结构、并行结构、顺序循环结构、并行循环结构的结构方式组合在一起，形成一个逻辑单位，作为用于协同的移动 Agent 的行为规约。将开放环境下的方法调用需求与连线机制加以分离，用基于移动 Agent 的灵巧连线来代替传统的连线，并开发了若干应用[121~123]。

表 3.1　开放协同软件模型结构

	协同程序设计部分	软件实体部分
面向对象程序模型	指定被调用者的、同步紧耦合的、功能实现固定的方法调用	构成程序的对象间无规范结构的关联关系
开放协同软件模型	内置式、可演化的体系结构	软件服务的表现形式
	多模式共存的灵活交互方式	量大面广的服务提供
	面向协同的服务剪裁与增强	经常不断的新老更替

为了解决 M 所表示的交互模式的多样性问题，文献[24]提出了多模式交互机制及基于 Agent 中间件模型：首先对现有各种网络环境下的交互模式进行系统分析，分离出构成交互模式的各种要素及其组合方案，在一致性和完整性原则的指导下，提出一个交互模式分解/综合配置模型，如表 3.2 所示。

表 3.2　交互模式分解/综合配置模型

交互要素	关注要素	
	关注点	取　值
发送方	时间配置	无须返回、延时返回、即时返回
	空间配置	单一实体、中介实体、组合实体
	可信需求配置	有/无
	参数转换配置	有/无

续表

交互要素	关注要素	
	关注点	取　值
接收方	接收范围配置	无条件、基于角色、基于信任度
	功能剪裁与增强配置	有/无
	调用匹配策略	结构匹配、内容匹配
参与媒体	信息传递方式	消息传递、代理传递
	安全需求配置	有/无

在此基础上，借助自省中间件技术，设计了基于截获者（Interceptor）设计模式的开放中间件模型，并开发了应用平台[124~126]。

针对关联关系 O 的结构化问题，文献[24]提出了面向体系结构的协同程序设计方法：扩展了面向对象程序模型中对象的理念，引入体系结构对象的概念，形成了内置式软件体系结构，并支持远程对象引用的重解释；通过体系结构对象的演化，利用对象系统的类型和多态机制，支持非预设的动态重配置；将动态体系结构实现为一组协同 Agent，借助其灵活的分布式模式，实现了体系结构分布共享。

基于上述研究成果，文献[24]提出了一条建立基于 Agent 的网构软件模型的技术途径：网构软件模型＝开放协同模型＋环境驱动模型＋智能可信模型。虽然从总体上看，该研究是从宏观的角度对网构软件进行分析和建模，并偏重于开放协同模型的设计，对构件或实体的自适应和自演化特征支持较少，但它为网构软件模型的研究奠定了良好的基础。

在前人研究的基础上，常志明等在文献[25]中提出了以 DAgent 为

构件的网构软件实体模型。文献[25]中的 DAgent（Dynamic Agent）是一类特殊的 Agent，除具有一般 Agent 的自主性以外，它还可以通过动态调整其内部的行为规约，在变化的环境中展现出自适应行为。其核心是通过将 DAgent 动态绑定到角色（Role）的方法来实现 DAgent 的动态行为，同时在 Role 中利用 EBDI（Electronic Business Document Exchange）结构显式定义环境信息，以对自主行为进行规约和实现。EBDI 结构如图 3.1 所示。

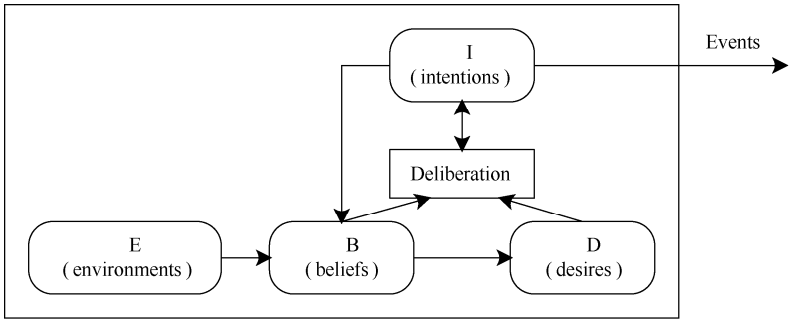


图 3.1　Role 中的 EBDI 结构

文献[25]的 EBDI 模型包含了 4 个基本组成部分：B 为信念，是 Agent 对自身的理解和认知，是 Agent 进行动作决策的基础；D 为期望，反映了 Agent 可能试图实现的任务和目标，体现了 Agent 的某种意向；I 为意图，是对未来动作的合理选择，它将影响和约束 Agent 的行为决策和实施，是 Agent 动作的起因；环境 E 是外部可见的，能影响实体的内部状态，并且实体动作的执行也会对环境产生影响。文献[25]中还通过 Role 模型的形式化定义，描述了构件的运行状态、自主运行及自适应演化运行机制。

3.1.2　基于自主构件的网构软件实体模型

目前，构件技术已经得到了很大程度上的认可，产业界支持和推动了基于构件的软件开发，特别是基础设施方面的建设，如构件库、构件运行支撑平台等都已经有很多实用的产品[127,128]。文献[22]和[23]在现有构件技术的框架下对构件进行改进，将 Agent 领域在自主性上的研究成果集成到传统的构件模型之中，形成了具有自主性的构件，称为自主构件。其结构如图 3.2 所示。

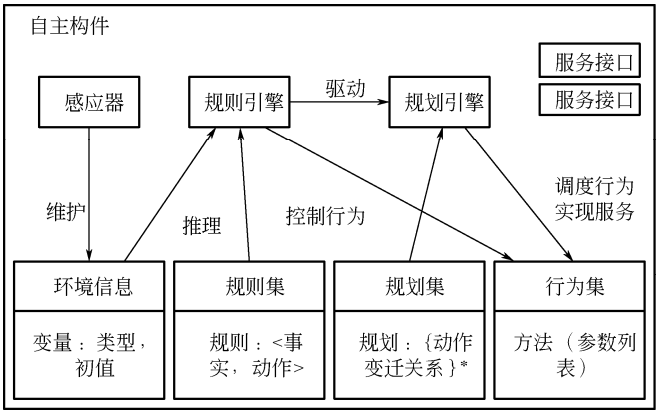


图 3.2　自主构件实现结构

文献[22]中的自主构件包括以下部分：环境信息刻画了自主构件所关心的并能感知到的各类环境变量的相关信息；感应器用来感知外界环境的状态变化，并根据环境的状态维护环境信息；行为集用于保存业务逻辑的计算方法；规则集是驱动和约束自主构件行为规则的集合，通过“事实（fact）+动作（action）”的表示方法规定当给定事实为真时，构件应执行的相应动作；规划集中的规划描述了自主构件在提供服务时应该

采取的任务步骤，可通过在线方式或离线方式生成；规则引擎用于根据环境信息及规则集推理自主构件的行为。

文献[22]中定义的自主构件响应请求、提供服务的基本行为模式如下：

（1）当收到服务请求时，规则引擎根据当前的环境状态及规则集对自主构件的行为进行推理，如果不存在可触发的规则，则不采取任何行为；否则启动规划引擎，选择并执行相应的规划来实现服务。

（2）在提供服务的过程中，如果所有的动作都能正确执行，且不存在某个动作违反限制性规则，则根据规划所规定的步骤执行以提供相应的服务；否则，产生服务失败例外。

（3）即使在没有外界服务请求的情况下，规则引擎也可能会根据自主构件所感知到的环境状态信息来触发行为规则，执行特定的行为，以调整其内部状态。

3.1.3 支持软件自适应的网构软件实体模型

对环境的适应是软件保证其可信的重要手段，文献[26]提出了适应性构件模型（Adaptive Component model for Open Environment，ACOE），实现感知、决策和执行三者的关注点分离，以保证网构软件实体对环境适应能力的在线调整。ACOE 主要包括三方面内容。

（1）文献[26]中为 ACOE 定义了两种构件类型：行为构件和感知构件。行为构件是对具体业务行为进行封装的可组装软件单元，感知构件是对环境感知手段进行封装的可组装软件单元。其定义如下：

ACOE 构件类型::=<行为构件，感知构件>

行为构件::=<所提供服务接口集合，所依赖服务接口集合，依赖的上下文端口集合，构件实现体>

感知构件::=<所提供上下文端口集合，所依赖上下文端口集合，构件实现体>

上下文::=<名称，类型>

（2）ACOE 中通过 3 类连接子对构件间交互机制进行独立封装：服务连接子，用于绑定服务接口；上下文连接子，用于建立上下文事件通道；策略连接子，用于封装适应决策逻辑。各连接子定义如下：

ACOE 连接子类型::=<服务连接子，上下文连接子，策略连接子>

服务连接子::=<服务提供者行为构件，服务提供者接口，服务消费者行为构件，服务消费者接口>

上下文连接子::=<消费上下文事件构件，事件消费者端口，提供上下文事件构件，事件提供者端口>

策略连接子::=<提供上下文事件构件，事件提供者端口，环境模型上条件，方法调用序列>

（3）文献[26]中还定义了 AcoeADL 软件体系结构描述语言，用于描述由构件和连接子组成的系统蓝图，使用容器来实例化该蓝图，通过软件体系结构的在线修改来实现可信演化。ACOE 通过体系结构修改规约、默认约束和用户自定义约束来维护软件体系结构在线修改时的完整性。

3.2　强可信智能实体模型

基于 Agent 的网构软件实体模型可以很好地实现网构软件实体所具有的相对独立性、主动性和自适应性特征，但也存在一些不足。首先，

目前软件 Agent 技术尚不成熟，缺少公认的 Agent 定义，对 Agent 开发和运行平台的研究缺少工业化的实现；其次，互联网上的软件实体并不都是软件 Agent，对 Agent 互操作协议等的支持有限。因此，单纯基于 Agent 技术来开发网构软件实体会在使用范围上受到一定程度的制约。自主构件作为对普通构件的改装，将构件技术与 Agent 领域的自主技术结合起来，能够扩展业界主流的构件技术、复用已有构件和运行基础设施，具有更大的应用范围。前两种模型实现了网构软件模型中的开放协同模型和环境驱动模型，但对于智能可信模型的研究还有所欠缺，ACOE 模型则提出了相应的解决方案。

纵观这三种模型，在智能可信方面还存在以下三方面不足。

首先，这些模型都通过自身内部结构与行为的演化实现了模型的自主性，但对于可信性保障机制没有明确的演化策略，或者说将可信性的演化与功能的演化掺杂在一起进行，这样不利于软件可信性的保障。

其次，这些模型重点实现了通过感知外部环境来进行相应的行为，但对于实体内部可信性变化情况监控不足。

最后，这些模型都是通过反射机制向外界反映实体自身的信息，这对于可信性保障的实现还是不够的。

本书在构件概念的基础上，提出了网构软件强可信智能实体模型。该模型具有自省性、自明性、自主性，可以实时监控自身的可信性水平，并通过特定接口向外界公布以供验证，而且可以感知环境的可信变化情况而对自身的可信性行为做出自主演化，以便能够适时地响应。这种自说明、可评估、能演化的强可信实体，为实体间信任建立了基点。本节首先阐述网构软件可信实体模型应具有的特点，然后对模型的结构与行为进行设计。

3.2.1　网构软件可信实体模型的特点

作为传统软件结构的自然延伸，网构软件具有区别于传统软件形态的独有的基本特征，包括自主性、演化性、协同性、多态性和反应性等[129]，具体表现在：网构软件实体可在不同的网络节点上独立运行，其目标和所提供的服务由其所有者来决定，其行为受自身的目标驱动，而并非单纯地被动用于组装或部署；在运行过程中以某种方式暴露自身状态和行为信息的同时，能够实时收集环境的各种变化信息，并根据预先设定好的策略，在必要时自动调整自身的行为以适应环境的变化；软件实体之间的连接方式可根据网络环境做出适应性调整；软件体系结构具备动态调整能力；软件系统具有多目标，并能够在运行时基于环境变化进行动态目标的适应性选择。

在可信性保证方面，也应遵循网构软件实体所具有的特征。第一，可信性的保证应具有确定的目标，网构软件实体应明确须保持在怎样的可信性状况下才能实现自身的功能目标；第二，网构软件实体应明确自身目前的可信性情况，以此作为调节自身行为的依据；第三，网构软件实体需要与外部环境中的其他网构软件进行协同，因此需要将自身可信性变化情况向外界发送；第四，网构软件实体还需要根据外界可信性变化情况适时调整自身行为，以保证自身目标的实现。

在这里引入可信计算中信任链的思想，即网构软件中的各个实体都成为有明确的可信性保障机制的可信实体，再通过实体间的信任约束建立起信任链，使信任链上的各实体在自身可信的基础上保证交互可信，以达到系统整体的可信。因此，本书所提出的强可信智能实体模型应具

有自省性、自明性、自主性的特点。

自省性是网构软件实体可信的基础，要求实体能够实时监控自身运行状况，及时发现自身可信性下降情况，并根据预设策略进行相应处理。由于实体可能需要向外界环境中的其他实体提供服务，自省性可以保证自身所提供服务的可信性，以此为基础建立实体间可信交互。

自明性是网构软件中实体间建立可信交互的途径，要求实体在与其他实体通信时显式声明自身可信性的情况，同时在自身可信性发生变化时及时向外界发布信息，以使其他实体能够根据此情况及时做出反应。此外，实体自身的可信性保障情况也可向其他实体发布以供评估，但应注意细节的保密性，以免被攻击者所利用。

自主性是实体及实体集合向外界提供可信服务的保障，要求实体可以根据环境状态及其变化，实时调整自身可信性保障行为。由于网构软件的多态性，软件系统具有多目标，实体的可信性保障行为也受自身目标驱动。实体需要根据目前可信性状况调整目标实现方式及服务的提供时机、方式等。

自省性和自明性相结合构成了建立可信交互的基础，保证了实体及实体集合所提供服务的“强可信”；自主性保证了实体能够根据外界情况自行决定服务方式以达到系统效率的最大化，满足了对实体自主、协同的要求。这样就为实体自身的可信性建立了保障。

3.2.2 模型结构设计

模型结构在总体上分为业务功能模块和可信性保障模块。业务功能模块与可信性保障模块在构成上是相互分离的，可信性保障模块成为

显式的独立实体，不对业务功能的执行造成任何干扰。可信性保障模块可以随时从实体结构中分离出来，允许实体动态开启或关闭可信性保障功能，动态配置和调整可信性保障需求，动态选择不同的可信性保障模块，从而实现可信性保障的动态性和可控性。在应变上，业务功能模块主要通过多种构件的组合、协同来为用户提供多样性、个性化的服务；而可信性保障模块则通过对内部构件运行情况和外部实体间交互事件进行监控，以可信性保障策略的演化来适应内、外部情况的变化。强可信智能实体模型如图 3.3 所示。

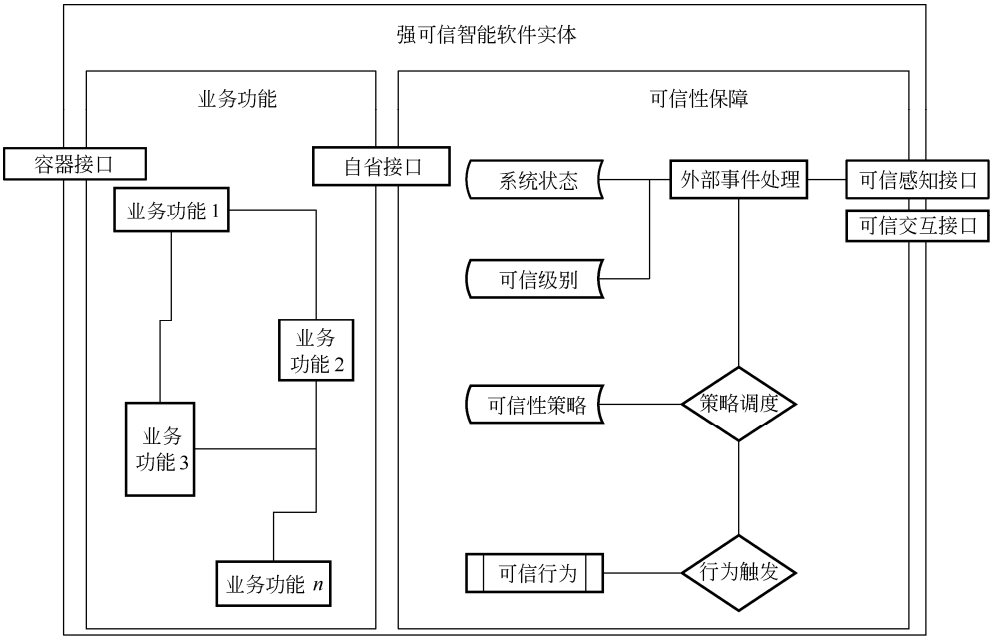


图 3.3　强可信智能实体模型

实体模型分为业务功能模块与可信性保障模块两部分。业务功能模块包含实体自身的业务逻辑，通过容器接口与实体容器相连，向外界提供正常的业务服务。可信性保障功能从业务功能中独立出来，形

成单独的可信性保障模块，并与业务功能隔离，只通过自省接口进行单向访问。可信性保障模块主要负责与其他实体进行可信性交互，并处理从外部获取的可信性相关事件。可信交互接口是实体向外界发布自身可信性信息及获取其他实体可信性信息的通道。可信感知接口则用于获取实体容器激发的可信性事件，由外部事件处理构件根据系统当前状态确定系统可信级别，并按照预先定义的可信性策略触发系统可信性行为。

1．自省性设计

实体需要监控自身业务功能的运行以发现运行过程中导致可信性降低的风险因素，以便采取保障措施进行恢复。实现自省的前提是业务功能与可信性保障的分离，由可信性保障模块通过自省接口对业务功能进行保障与监控。这样，当实体发现自身可信性情况发生变化时，可以应用可信性保障策略主动改变自身的行为以做出响应；同时，当业务功能发生变化时，只需要对业务功能模块内部构件进行调整，而不会影响可信性保障；另外，同一套可信性保障机制也可以应用于不同的业务功能中，提高了模块的复用性。实体的自省通过静态与动态两种方式进行。

（1）静态自省：抽取实体自身业务模块的运行及交互逻辑，通过形式化的方法对其进行描述。将形式化的描述保存在实体配置信息中，并由可信性保障模块对描述信息进行计算与验证，对其可靠性、可用性、效率等方面的情况进行评估。当业务逻辑发生变化时，需要重新执行“抽取—存储—评估”的过程。

（2）动态自省：实时动态监测实体的运行情况，收集实体行为信息，及时发现实体异常或实体间的异常交互，以及外界可能对实体进行的攻

击行为，为实体自身的可信策略及自主行为提供可用的指导信息。

通过自省机制，实体可以保障自身的高可信，为向外界承诺自身的可信性提供了良好的基础，从而建立了实体间信任的基点。

2．自明性设计

自明性的实现需要建立在实体自身可信性保障机制能够被其他实体验证的基础之上，同时实体自身的核心业务细节需要具有保密性，不能被其他实体探知。因此，实体模型的设计同样需要将实体的业务功能模块与可信性保障模块分离开，使可信性保障模块可以通过可信交互接口与外界实体进行交互，而隔离功能模块的业务逻辑细节。实体可以将保存在配置信息中的自身业务逻辑形式化描述，以及实体自身的可信性保障信息，通过可信交互接口发送给欲进行交互的目标实体，供目标实体进行验证，做到自身可信性状态的承诺，为信任契约的建立奠定基础。另外，当实体需要其他实体提供服务时，也会通过可信交互接口获取多个具有相同服务功能的实体的信息，根据可信性评估选择可信性最高的实体进行交互。

自明性还可以向外界提供对服务实体自身的描述。服务描述是服务组织和访问的基础，要实现对不同服务之间的通用访问，要求服务描述中提供两方面的信息：①服务的能力信息，即说明服务“能够做什么”，以支持其他实体的一致性访问，主要包括功能类规范、服务自身可信性的说明等；②服务的行为信息，即说明“怎样使用服务”，通常由一个或多个接口表示，主要包括各功能调用接口、可信性声明接口、可信性监控接口等。所有的说明信息与交互接口都通过形式化的方式予以描述，服务运行过程中都可对其进行验证，从而建立信任传播的良好途径。

需要指出的是，自明性的设计并不同于编程语言中的反射系统，主要表现在以下三个方面。

（1）从使用目的方面，反射系统是对应于软件工程中的抽象封装而产生的，以开放实现为目的，通过暴露模块的实现细节来向用户提供系统内部的相关信息。而位于开放互连网络中的网构软件实体都是由各开发商自行开发维护的，其中不仅包含了开发商的知识产权，还涉及运行环境的安全问题，因此不可能向外界公布具体的实现细节，只能通过可信交互接口以描述语言的形式向外界提供自身的能力信息、行为信息、可信性保障信息等，供使用者进行验证评估，以达到自明的目的。

（2）从交互机制方面，反射系统主要是在面向对象系统中通过方法调用来实现对目标对象内部细节的访问。而网构实体是松散耦合的，分布在不同的网络节点上，实体的实现方式也往往是异构的，因此它们之间的交互手段不能像普通构件那样基于简单的方法调用，而需要通过统一协议下的描述机制来实现。

（3）从操作对象方面，反射系统应用于对象之间的操作。而自明性的主体——网构软件实体，则可能是由若干对象组成的服务，甚至是一个完整的软件系统。

3.2.3 模型行为设计

网构软件的实体之间可以以动态或静态的合作方式在网络环境下互连、互通、协作，进而实现并完成一定的功能和任务。因此，实体需要通过专门的接口监测外界信息，以感知外界环境发生的变化。同时，实体内部需要具备外部事件处理机制，以便适时采取适当的可信性策

略，在外界可信性环境发生改变时，主动改变节点的行为，使得在满足功能需求的同时能满足实体的可信性要求。采用如图 3.4 和图 3.5 所示的系统结构建立实体自主性设计模型。

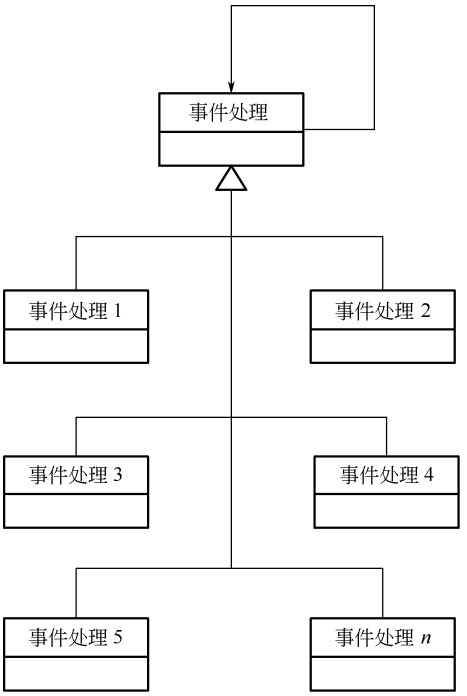


图 3.4　外部事件处理模型

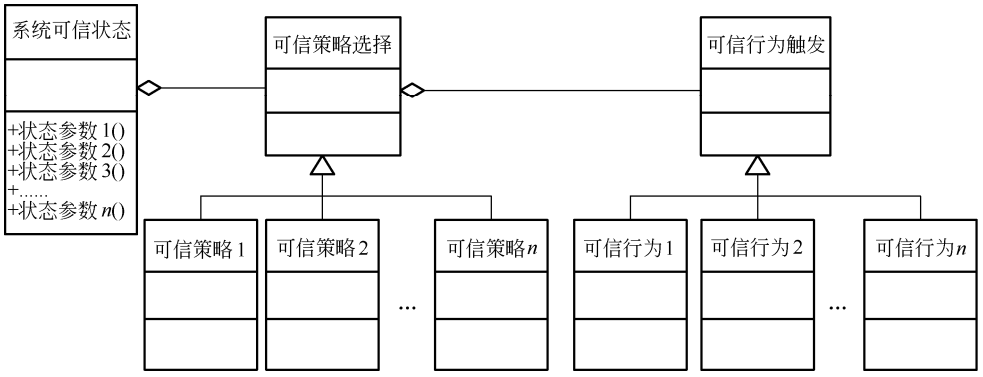


图 3.5　策略行为调度模型

自主演化过程如下：

（1）外界环境感知。通过可信性感知接口监测外界环境可信性的变化状况，当监测到发生可信性变化的事件时，事件处理构件捕获外部事件并进行处理。通过职责链的方式建立如图 3.4 所示的事件处理总线，外部事件沿总线传递，各事件处理构件都可以通过总线捕获自身可以处理的事件，直到有一个构件进行处理为止。这使得总线上的每个构件都有机会处理事件，从而避免了具体事件与固定处理构件的耦合关系，并且可以根据外界情况非常方便地添加、删除和替换事件处理构件，以保证实体的动态演化。

（2）系统可信状态设置。系统可信状态用于标识系统目前可靠性、可用性、效率等各项可信性指标所处的状态，通过该状态参数为实体提供外界环境的可信性情况。事件处理构件对外部事件进行处理时，会根据处理情况设置系统的状态，以作为系统进行策略调度的依据。同时建立系统的可信性级别，一个状态级别由一系列的系统状态值组成，状态值既表明当系统处于该可信性级别时所监控的外界环境要处于具体状态参数所设定的范围，也表明实体在响应某一服务请求时希望外界应该达到的可信性状态，并以参数形式具体反映。

（3）可信性策略调度。如图 3.5 所示，事件处理构件通过一系列的系统状态值，以及系统可信性级别发生改变的情况，选择合适的可信性策略，以触发实体的自主行为，实现实体的自主服务。可信性策略是预先定义的一系列算法，并将其封装在不同的策略构件中，使其可以相互替换，并独立于系统状态及事件处理构件而变化。实体进行动态演化时，可以根据实际情况对现有策略执行添加、删除和替换操作。

（4）可信性行为触发。如图 3.5 所示，当外界环境可信性状态改变时，根据预先定义的可信性策略，触发实体的自主行为，以确保实体的可信性要求，且不同状态间转变要触发不同的行为操作。当可信性行为被触发时，通过自主行为调整实体本身的服务状态及可信性保障手段，实现自身演化，以确保自身的可信性能够满足要求。

（5）系统实时状态交互。通过实体自主行为调整自身服务或采取新的可信性保障措施后，需要将变更后的功能服务情况通过可信交互接口发送至存在交互关系的其他实体，以使其做好准备。同时，实体可信性保障措施以及实体自身的可信性变化情况，也需要通过可信交互接口向外界发布，以使其他实体能够对此进行评估。

3.3　实体可信情况形式化描述语言

体系结构描述语言是参照传统程序设计语言的设计和开发经验而设计的针对软件体系结构特点的专门的描述语言。它在吸收了传统程序设计中语义严格精确的特点基础上，针对软件体系结构的整体性和抽象性特点，定义和确定适合于软件体系结构表达与描述的有关抽象元素，包括构件、连接子等。使用这些元素可以很好地描述网构软件内部各构件间的相互关系及行为，既可为自身可信性情况监控提供良好基础，又可作为与外部环境中的其他实体进行协同的有效途径。本节阐述体系结构描述语言的概况，并在此基础上设计一种实体可信情况形式化描述语言（Entity Dependability State Architecture Description Language，EDSADL）。

3.3.1 体系结构描述语言

体系结构描述语言[130]（Architecture Description Language，ADL）是对软件体系结构的一种形式化的描述方法，它提供了具体的语法与刻画体系结构的概念框架，使得系统开发者能够很好地描述他们设计的体系结构，以便与他人交流，并且能够用提供的工具对许多实例进行分析。Paul C. Clements[131]在 1996 年对当时的 ADL 现状和发展进行了总结。随后，ADL 快速发展，相继出现了多项研究成果。N. Medvidovic 和 R.N.Taylor[132]在 2000 年对已有的 ADL 进行了全面的分析和总结，提出了一个评价框架，并对已有的 ADL 进行了评述，包括 Aesop，Artek，C2，Darwin，MetaH，Rapide，SADL，UniCon，Weaves，Wright 和 Acme 等多个系统。另外，我国研究人员也在 ADL 的研究中取得了一些成果，提出了 XYZ/ADL[133]和 ABC/ADL[134]。

虽然目前对于 ADL 的研究已经取得了大量成果，为软件的设计与开发提供了良好的帮助，但现有的 ADL 都存在一定的局限性，主要表现在以下方面。

（1）每个 ADL 系统都是针对特定项目开发的，有一定适用范围，目前还没有一个 ADL 系统可以做到通用。

（2）已有的 ADL 主要解决软件系统的架构问题，所以其重点在于对不同系统的模式设计，很少有对于可信性说明的设计。

（3）已有的 ADL 解决系统的设计问题时，只考虑构件或系统的复用，而网构软件系统的可信性保障，需要解决实体内部各构件可信性变化及实体间信任演化问题，以及这些变化对实体所提供服务造成的影

响，所以系统的重点在于对实体及其内部构件之间交互可信性的描述。

本节定义一种描述网构软件实体可信情况的体系结构描述语言 EDSADL。网构软件实体可以通过 EDSADL 描述自身构件可信性状况以及构件之间交互情况，追踪导致可信性降低的问题源，作为自省机制的基础并为自主演化提供支持；还可以根据描述信息及时向外界发布服务的可信性变化情况，为自明机制提供途径，满足对实体自省性、自明性、自主性的要求。

EDSADL 也是一种 ADL，也应具备 ADL 的某些特征。N. Medvidovic 和 R.N. Taylor[132]给出了设计 ADL 的基本框架，包括设计 ADL 时应具备的各种属性和功能，但没有给出具体元素的模型实现，即其中应包括的基本元素和各个元素的模型结构。

ADL::=<体系结构建模特征，工具支持>
体系结构建模特征::=<构件，连接子，体系结构说明>
构件::=<接口，类型定义，语义说明，行为约束，属性演化，非功能特性>
连接子::=<接口，类型定义，语义说明，行为约束，属性演化，非功能特性>
体系结构说明::=<可理解性，组合性，精化和可跟踪性，异构性，可伸缩性，演化性，动态性，约束，非功能特性>

David Garlan 在设计 ACME[135]时给出了 ADL 应具备的基本元素，包括构件（Component）、连接子（Connector）、系统（System）、属性（Properties）、约束条件（Constraints）、系统模式（Styles）六个基本元素。

3.3.2　EDSADL 元素的设计与描述

EDSADL 是为描述网构软件实体内部服务可信性情况而定义的体系结构描述语言，其目的是描述实体提供服务时内部可信性变化情况，

并向外界提供信息说明。而可信性的变化主要是由各构件的行为引起的，也是通过行为的监控而获取到的，更是以行为的结果为主要体现。因此，EDSADL 在 ADL 基本元素的基础上加入了对交互行为的描述，并以此为主线来描述实体各组成元素的可信性及其变化情况。EDSADL 包括构件、连接子、系统组成结构、构件间约束、系统模式、交互行为六个元素。

EDSADL 不但作为设计时的描述语言，还作为运行时构件及实体间信息交互的途径，因此不但要人工可读，还要机器可读。最好的方式就是采用 XML 描述。XML[136]作为一种网络标准，以一种开放、自描述的方式定义数据结构，可以同时描述数据内容和结构特性，并能够简化互联网上的数据交换。目前已有多种多样的解析工具用于软件系统对 XML 的生成、获取、解析、处理。本书的模型描述采用 XML Schema[137]定义 EDSADL 中各元素的各种属性，以便于系统运行时对产生的 EDSADL 文档进行实时处理。

1．构件（Component）

构件是体系结构中的一个计算或数据存储单元。其粒度可以小到一个算法模块，也可以大到一个完整的系统。构件通过接口与外部进行数据交互，可通过多个接口来处理不同层面的信息。在 EDSADL 中，构件被定义为一个语义完整、语法正确、可复用的单元。构件模型由接口、类型、可信性信息和实现机制组成。实现接口描述构件对外提供的功能及运行所需的功能支持，是与外界交互的基础。语义接口可用来提供外界对构件的识别信息，包括构件标识、语义描述等。构件类型说明构件是否向所属实体之外的其他实体提供服务，以便于实体对自身服务可信性的监测。可信性信息包括实体当前可信性值及可信性保障机制说明，

为实体自省与自明提供基础数据。EDSADL 中的构件描述如下：

```
构件::=<实现接口，语义接口，构件类型，可信性信息>
实现接口::=<提供功能接口，需求功能接口>
提供功能接口::=<参数列表，前置条件，后置条件>
需求功能接口::=<引用类型，提供参数>
语义接口::=<标识信息，语义信息>
可信性信息::=<可信值，可信性机制>
<xs:element name="Component">
    <xs:complexType>
        <xs:element name="ImplementInterface">
            <xs:complexType>
                <xs:element name="ServiceInterface">
                    <xs:complexType>
                        <xs:element name="Parameter"/>
                        <xs:element name="PreCondition"/>
                        <xs:element name="PostCondition"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="RequireInterface">
                    <xs:complexType>
                        <xs:element name="Type"/>
                        <xs:element name="Parameter"/>
                    </xs:complexType>
                </xs:element>
            </xs:complexType>
        </xs:element>
        <xs:element name="SemanticInterface">
            <xs:complexType>
                <xs:element name="ID"/>
                <xs:element name="Semantics"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="TrustworthyInformation">
            <xs:complexType>
                <xs:element name="Trustworthiness"/>
```

```
                    <xs:element name="TrustworthyMechanism"/>
                </xs:complexType>
            </xs:element>
            <xs:attribute name="Type">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="external"/>
                        <xs:enumeration value="internal"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:complexType>
    </xs:element>
```

按照构件功能的不同，将功能相似的构件集合在一起，形成构件子集合（ComponentSet），不同的构件子集合在构成实体的结构中处于不同的层，各层之间互相提供和需求服务。EDSADL 中的一个构件子集合包括了构件层名称和子集合中包含的构件。

```
<xs:element name="ComponentSet">
    <xs:complexType>
        <xs:all>
            <xs:element name="LayerName"/>
            <xs:element name="Components"/>
        </xs:all>
    </xs:complexType>
</xs:element>
```

2．连接子（Connector）

连接子是用来建模构件间交互及其控制规则的模块。在体系结构中，连接子可以是一个单独的消息处理模块，也可以以构件之间数据交互的形式存在，如通信协议、缓存、队列、管道、SQL 连接等。连接子

同样有接口定义，为不同构件的操作提供控制角色。EDSADL 的连接子模型，主要包括交互角色、行为特性、可信性信息三部分。交互角色定义了交互构件之间所期望的行为，规范了参加连接的构件职责。通过设定不同的角色，定义了构件之间的操作规则，增强了构件之间操作的可追踪性，有助于追溯错误发生源。行为特性定义了构件交互过程中的部分特性，包括行为激发顺序、该连接所允许的最大用户数量、交互方式（同步或异步）、构件交互所使用的协议等。可信性信息包括实体当前可信性值及可信性保障机制说明。EDSADL 中的连接子描述如下：

```
连接子::=<交互角色，行为特性，可信性信息>
交互角色::=<构件行为 1，构件行为 2>
行为特性::=<激发顺序，用户数量，交互方式，交互协议>
可信性信息::=<可信值，可信性机制>
<xs:element name="Connector">
    <xs:complexType>
        <xs:element name="Properties">
            <xs:complexType>
                <xs:attribute name="RequestOrder">
                    <xs:simpleType>
                        <xs:restriction base="xs:string">
                            <xs:enumeration value="sender to receiver"/>
                            <xs:enumeration value="receiver to sender"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="MaxUser">
                    <xs:simpleType>
                        <xs:restriction base="xs:integer"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="Synchronous">
                    <xs:simpleType>
```

```
                        <xs:restriction base="xs:boolean"/>
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="Protocol">
                    <xs:simpleType>
                        <xs:restriction base="xs:string"/>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </xs:element>
        <xs:element name="TrustworthyInformation">
            <xs:complexType>
                <xs:element name="Trustworthiness"/>
                <xs:element name="TrustworthyMechanism"/>
            </xs:complexType>
        </xs:element>
        <xs:attribute name="Role">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="sender"/>
                    <xs:enumeration value="receiver"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:element>
```

3．系统组成结构（SystemStructure）

系统组成结构描述构件和连接子通过怎样的拓扑结构来组成系统。系统中的所有构件可根据所提供功能的不同划分为多个子集合，依据其具体的功能类型定义为不同的构件层。因此，系统组成结构又包含三个方面：系统整体结构，描述构件在整个系统中所处的位置；构件层次关系，描述各构件层之间的关系；构件语义关系，描述处

于同一构件层的构件之间的关系。这三种结构分别描述了构件三个层面的特征，由此可以自底向上由原子构件追溯到所有此构件涉及的服务，也可以自顶向下由服务查找到所有相关的构件。

```
系统结构::=<系统整体结构，构件层次关系>
系统整体结构::=<构件子集合，连接子集合，组织规范>
构件层次关系::=<构件语义关系，相关层名>
构件语义关系::=<构件名，所属子集合名，交互关系>
交互关系::=<单向调用|双向交互>
单向调用::=<方向标识，构件语义关系，指向元素>
双向交互::=<方向标识，构件语义关系，指向元素>
<xs:element name="SystemStructure">
    <xs:complexType>
        <xs:element name="Structure">
            <xs:complexType>
                <xs:element name="ComponentSet"/>
                <xs:element name="ConnectorSet"/>
                <xs:element name="Specification"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="LayerStructure"/>
    </xs:complexType>
</xs:element>
<xs:element name="LayerStructure">
    <xs:complexType>
        <xs:element name="Relation">
            <xs:complexType>
                <xs:choice>
                    <xs:element name="One-way">
                        <xs:complexType>
                            <xs:attribute name="Direction"/>
                            <xs:attribute name="ComponentName"/>
                            <xs:attribute name="ComponentLayer"/>
                            <xs:attribute name="Next"/>
                        </xs:complexType>
```

```
                    </xs:element>
                    <xs:element name="Multi-way">
                        <xs:complexType>
                            <xs:attribute name="Direction"/>
                            <xs:attribute name="ComponentName"/>
                            <xs:attribute name="ComponentLayer"/>
                            <xs:attribute name="Next"/>
                        </xs:complexType>
                    </xs:element>
                </xs:choice>
                <xs:attribute name="ComponentName"/>
                <xs:attribute name="LayerName"/>
            </xs:complexType>
        </xs:element>
        <xs:attribute name="RelatedLayer"/>
    </xs:complexType>
</xs:element>
```

4．构件间约束（SemanticMap）

构件间约束是对系统或其组成部分的属性或行为的断言，在系统的演化中保持正确。EDSADL 中定义各构件间的责任与义务为构件间约束。在系统演化中，可保证系统功能语义的正确性，同时可在出现故障时发现故障源。设构件子集合中包含 n 个构件，则定义二维的 $n \times n$ 的表结构来表示此构件子集合中各构件之间的约束关系。构件之间的语义关系是有方向的，行为语义关系的末尾构件，列为语义关系的起始构件。此结构为可信性问题的溯源提供了规则。

```
<xs:element name="SemanticMap">
    <xs:complexType>
        <xs:all>
            <xs:element name="Table">
            {a_ij}_n*n<!--当第 i 个构件到第 j 个构件有一个顺序规则时，a_ij=1-->
```

```
            </xs:element>
        </xs:all>
        <xs:attribute name="ComponentSetName">
            <xs:simpleType>
                <xs:restriction base="xs:string"/>
            </xs:simpleType>
        </xs:attribute>
    </xs:complexType>
</xs:element>
```

5．系统模式（Styles）

系统模式代表一类风格类似的体系结构，定义了一系列构成此类体系结构的基本元素及其类型、组成规则、结构框架等，以规范应用系统的实现。由于目前网构软件系统的构建多采用多层次的服务组合模式，因此在 EDSADL 中只定义了一种模式，即分层结构模式。

```
系统模式::=<构件子集合，连接子集合，系统组成结构，构件间约束，系统规范>
<xs:element name="Styles">
    <xs:complexType>
        <xs:choice>
            <xs:element name="MultiLayer">
                <xs:complexType>
                    <xs:all>
                        <xs:element name="ComponentSet"/>
                        <xs:element name="ConnectorSet"/>
                        <xs:element name="SystemStructure"/>
                        <xs:element name="SemanticMap"/>
                        <xs:element name="Specification"/>
                    </xs:all>
                </xs:complexType>
            </xs:element>
        </xs:choice>
```

```
        </xs:complexType>
    </xs:element>
```

6．交互行为（Activity）

交互行为用于描述系统提供服务时的动态行为，每次服务形成一个行为对象，其中包含了所涉及构件间的相互操作，为系统动态监控自身行为情况提供了基础。监控机制捕获此行为信息后，可解析行为成功或失败的情况，并分析导致失败情况的根源。同时，行为的具体情况还可反映出各构件的当前情况，为构件选择及系统演化提供依据。根据分析结果，系统可相应地提高或降低相关构件或服务的可信性值，并根据此数值进行下一步的可信性保障活动，如改变自身行为或向外界发出通知等。

```
行为::=<服务构件，客户构件，连接子，交互情况>
交互情况::=<交互成功|交互失败>
交互失败::=<报错源，目标构件，错误类型，出错信息>
<xs:element name="Activity">
    <xs:complexType>
        <xs:all>
            <xs:element name="ServerComponent"/>
            <xs:element name="ClientConponent"/>
            <xs:element name="Connector"/>
            <xs:element name="InteractiveInformation">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="Success">
                            <xs:simpleType>
                                <xs:restriction base="xs:string"/>
                            </xs:simpleType>
                        </xs:element>
                        <xs:element name="Fail">
                            <xs:complexType>
                                <xs:all>
```

```
                                        <xs:element name="Source"/>
                                        <xs:element name="Target"/>
                                        <xs:element name="Type"/>
                                        <xs:element name="Information"/>
                                    </xs:all>
                                </xs:complexType>
                            </xs:element>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
            </xs:all>
        </xs:complexType>
    </xs:element>
```

3.4　基于 EDSADL 的实体自省机制

为了保证动态演化性，网构软件的体系结构一般包括目标层与元层两个层次。目标层包含系统的主要功能，用于满足用户的各项需求；元层通过感知环境变化来对目标层进行演化，以调节系统行为。本节所描述的自省机制，通过监控的方法获取实体中各构件的可信性情况，为网构软件的元层提供可信性方面的演化依据。自省机制包括以下 4 个步骤。

1．基于关联矩阵的构件可信性描述

根据构件间的交互关系建立起实体组成结构的关联矩阵，如图 3.6 所示。

图 3.6 中描述了有直接交互的构件关系，其方向性由行和列分别表示。行为被关联方向的末尾构件，列为被关联方向的起始构件。矩阵中

有数值的元素表示对应两个构件之间存在直接交互关系，其数值表示交互成功的概率，即可信性。值得注意的是，两个构件之间不存在直接交互关系是通过矩阵中没有数值的元素表示的，而数值 0 表示对应构件间有交互关系且可信性为 0。

	C_1	C_2	C_3	C_4	C_5
C_1		0.98			
C_2				0.95	
C_3	0.9				
C_4			0.85		0.92
C_5			1		

图 3.6　实体组成结构关联矩阵示例

2．基于 EDSADL 的交互行为监测

针对需要监测的构件建立监测点，用以记录构件间交互情况。监测点的建立可使用 AOP 的动态织入机制，该机制目前已有许多研究成果[139,140]，可以在不改变目标构件源代码的情况下实现行为的监测，并能够根据监测需求动态调整监测构件或者开、关监测系统。

当监测到构件间交互失败的行为时，根据 EDSADL 中的行为定义、系统结构中构件关系定义、连接子定义，可将多个有直接交互关系的构件所产生的行为连接成一条错误链。沿此错误链向下追溯，可最终获得产生错误的行为源。设该交互关系发生的成功次数为 x，失败次数为 y，定义下次交互成功的概率即可信性为 $\hat{\theta}_{dt}$，则此概率服从 Beta 分布[141]，其密度函数为

$$\mathrm{Beta}(\theta \mid x, y) = \frac{\Gamma(x+y+2)}{\Gamma(x+1)\Gamma(y+1)}\theta^{x}(1-\theta)^{y} \tag{3.1}$$

且

$$\hat{\theta}_{dt} = E(\text{Beta}(\theta \mid x+1, y+1)) = \frac{x+1}{x+y+2} \tag{3.2}$$

其中 $0<\theta<1, x>0, y>0$

按照式（3.2）可计算出此交互关系的可信性并更新至关联矩阵。

3．基于规则的错误构件诊断

由于构件之间具有相互关联关系，某个构件出错会引起多个相关构件产生错误行为，因此可根据错误行为之间的关联关系判断错误源构件。定义规则如下：

```
错误关联规则::=<规则体，规则项>
规则项::=<源构件，目的构件，连接子，规则描述>
```

规则体用于定义规则可以应用的出错情况，规则项用于判断当前系统状态下是否可以执行选中的规则。可以根据多种不同类型的错误对规则项中的元素进行特别定义，并建立规则库。对监测到的行为应用规则库中的规则可得到错误源构件。例如，接受构件 A 服务的多个构件均产生行为错误，则判定构件 A 为错误源。

4．自省后续处理

后续处理分为内部与外部两方面。

内部处理：若某构件的可信性降低至低于预设阈值，则触发元层可信性相关事件，由元层执行对目标层的演化机制，替换相应构件。

外部处理：下层构件间的交互失败时，沿错误链向上追溯，若到达对外部提供的服务，说明该错误对服务的可信性造成了影响，则修改信任契约中的自身可信性情况说明，以向外界发出通知。

第 4 章

网构软件实体间信任约束机制

目前网构软件可信性保障的研究通常采用信任度量为网构软件提供的一种动态演化的、相对的、“柔性”的可信保障机制[75]，即选取信任度高的实体进行协作，并且只允许信任度较高的实体访问自身提供的服务。而在开放协同的环境下，很难为一个没有任何推荐者的全新实体设置默认信任值。而且实体间的信任关系易受到恶意推荐的影响，评估者很难在与目标实体交互前得知推荐者推荐信息的正确性。另外，信任传递过程中的衰减度也会因实体自身情况、所处网络状况等原因而各不相同。因此，本章通过引入契约式设计思想，采用“承诺—评估”机制建立基于契约的网构软件实体交互模型，保证交互双方可评估的可信关联关系。在此基础上提出基于评估的信任衰减过程，将主观评价与客观评估结合起来，使信任传递过程中信任的衰减参数的计算更加客观、准确。通过这样的显式约束机制，建立了信任链传递过程中的可信认证机制，与强可信智能实体相结合形成了信任链的整体结构。

4.1　网构软件的信任度量及演化模型

信任度量可为网构软件提供一种“柔性”、相对的软件度量机制，选取信任度高的实体进行协作，并且只允许信任度较高的实体访问自身提供的服务，这为网构软件可信性保障提供了支持[75]。文献[75]和[142]在互联网中实体间信任关系模型的基础上提出了一个适用于网构软件的信任度量及演化模型，并在此基础上提出了一种面向网构软件体系结构的信任驱动服务选取机制。

文献[75]认为，信任是软件实体关于其他实体或实体集团具有完成某一特定任务能力可能性的主观判断，其程度依赖于实体对于信任对象的直接经验和推荐信息；信任关系采用二元组 $R=<t,d>$ 描述，其中 t 表示可信度，d 表示不可信度，且 $t+d=1$；直接经验表示为二元组 $<s,f>$，s 为成功协作的次数，f 为失败的次数，其信任值为 $t=s/(s+f),t\in[0,1]$；推荐信息表示为 $R_{I:O}^{A\leftarrow C}=<t_{I:O}^{A\leftarrow C},d_{I:O}^{A\leftarrow C}>$，即实体 C 提供给 A 的关于实体 O 的推荐信息。

为了支持信任信息的传递与合并，文献[75]定义了两类算子。

（1）传递算子⊗：实体 A 为评估者，实体 C 为评估对象，实体 B 为 A 的一个推荐者，设 $R_{r:B}^{A}=<t_{r:B}^{A},d_{r:B}^{A}>$ 为 A 对于 B 的推荐信任，$R_{d:C}^{B}=<t_{d:C}^{B},d_{d:C}^{B}>$ 为 B 对于 C 的直接信任，则

$$R_{I:C}^{A\leftarrow B}=R_{r:B}^{A}\otimes R_{d:C}^{B}=<t_{I:C}^{A\leftarrow B},d_{I:C}^{A\leftarrow B}>$$

其中，$t_{I:C}^{A\leftarrow B}=t_{r:B}^{A}\times t_{d:C}^{B}$，$d_{I:C}^{A\leftarrow B}=t_{r:B}^{A}\times d_{d:C}^{B}+d_{r:B}^{A}\times t_{d:C}^{B}+d_{r:B}^{A}\times d_{d:C}^{B}$，即评估者得到的最终信任信息是由评估者对推荐者的推荐信任与推荐者对评估目标

的直接信任计算而得的，多层推荐者构成了一条信任链。

（2）合并算子⊕：评估者 A，A 的推荐者集合 RS，$r_i \in \mathrm{RS}$，$R_{r:r_i}^{A} =< t_{r:r_i}^{A}, d_{r:r_i}^{A} >$ 为 A 对 r_i 的推荐信任，r_i 所提供的推荐信息记为 $R_{I:O}^{r_i} =< t_{I:O}^{r_i}, d_{I:O}^{r_i} >$，评估对象 O，|RS|为集合中元素的个数，$R_{c:O}^{A}$ 为 A 关于 O 的合并信任，则

$$R_{c:O}^{A} = R_{I:O}^{A\leftarrow r_1} \oplus R_{I:O}^{A\leftarrow r_2} \oplus \cdots \oplus R_{I:O}^{A\leftarrow r_n} =< t_{c:O}^{A}, d_{c:O}^{A} >$$

其中

$$n = |\mathrm{RS}|$$

$$R_{I:O}^{A\leftarrow r_i} = R_{r:r_i}^{A} \otimes R_{I:O}^{r_i} =< t_{I:O}^{A\leftarrow r_i}, d_{I:O}^{A\leftarrow r_i} >$$

$$t_{c:O}^{A} = \sum_{i=1}^{n}\left(t_{I:O}^{A\leftarrow r_i} \times t_{r:r_i}^{A} \Big/ \sum_{j=1}^{n} t_{r:r_j}^{A} \right)$$

$$d_{c:O}^{A} = 1 - t_{c:O}^{A}$$

即评估者得到的最终信任信息是由多条信任链上的推荐信任合并而成的。

文献[75]使用请求—推荐模式处理复杂信任网络中信任信息的传递，即当评估者需要评估某实体的信任度时，向与其最近的一个或多个推荐者发送请求；而推荐者只有接收到推荐请求时才会将自身处理过的信任信息传递给请求者；当有多个推荐者可形成环路时，则推荐者只接受来自“高层”的推荐请求，以免造成死锁。

评估者根据最终获得的信任信息来决定是否与评估对象进行协同。每次协同后，评估者都会根据协同情况更新自身对于评估对象的直接信任。通过观测直接信任与推荐信息的相符程度来对推荐者的信任度做出评价。对于“善意”推荐者，提高其信任度；反之，降低其信任度。这样就完成了信任关系的演化。

在上述信任度量及演化模型的基础上，文献[142]提出了一种网构软件服务选取机制。在服务请求者与服务提供者之间引入一类被称为

“虚服务”的智能实体，用以替代传统体系结构中的连接子。虚服务按照服务类别连接多个外部服务，通过感知服务请求者的需求，动态选取、组合相应的服务，并在信任机制的驱动下进行替换与更新。为了描述服务请求者的需求及对虚服务的行为做出约束，文献[142]还定义了一种服务情境描述语言（Service Context Description Language，SCDL），包含表 4.1 所示的 5 类主要元素。

表 4.1　SCDL 的主要元素

元素	含　义
用户	调用网构软件所提供服务的人或客户端软件
需求	用户期望获得的服务调用结果，用于客户端接口语义信息描述
响应	描述特定需求的结果，用于服务调用端接口语义信息描述
计算实体	特定的网构软件系统、互联网中的各种软件服务和虚服务
行为	各类元素的一系列动作或关系

在 SCDL 的支持下，结合一阶谓词逻辑运算，可以对文献[75]中提出的信任演化策略建立描述算法，并以此作为虚服务选取、组合服务的基础，保证网构软件系统体系结构的顺利运行。

4.2　基于契约的网构软件实体间信任关系约束机制设计

作为软件实体交互过程中的双边规范，契约可分为从低层接口语义

定义到高层服务质量约束的多个层次。对网构软件实体来说，其交互方式不仅包含简单的消息传递与接口调用，还包含高级的通信语言及主动感知的通信手段。而用于约束网构软件实体间信任关系的契约也需要包含更丰富的信息，以便为实体间的协同、合作和竞争等提供有效评估依据。本节首先对契约式设计进行简要介绍，然后定义信任契约的描述方法。

4.2.1 契约式设计

契约[143]是一种多方（常见为双方）规范或约定，它规定了各参与方的权利和义务。在法律意义上，契约是双方及以上相互间在法律上具有约束力的协议。通常，契约责任是以自由同意为基础的，即契约自由原则。

契约式设计[144]（Design by Contract）是由 Bertrand Meyer 提出的一种软件设计方法学，通过将社会经济生活中的契约思想引入软件开发领域，从而确保开发出高质量的软件。对软件系统来说，只保证其各组成部分正确无误地运行是远远不够的，而最有价值的是在任何两个组成部分交接处设计明晰的彼此间的义务和权利规范，即所谓的契约[145]。

契约理论把一个软件系统当成一组相互通信的组件集合，而这些通信则建立在精确定义的双边规范或契约上。它提供关于通信建立的细节，尽管通信各方不需要知道对方的具体实现，但它们知道自己提供什么并需要什么。

目前，契约式设计已经在程序设计语言[146]、构件设计[147]、Web 服

务设计[148]等多方面得到了广泛的应用。

应用于构件或服务的契约可分为由低到高的 4 个层次[147]：语义契约、行为契约、同步契约和服务质量契约，可协商性依次增强。

（1）语义契约，是最基础的契约，用于支持最基本的系统运行。它通常用于定义构件所执行的操作、各构件所需的输入输出参数、构件执行时可能抛出的异常等。该层契约包括接口定义语言（IDL）、各种类型描述语言等，通过编译时的静态类型检查及运行时的动态类型检查来保证各构件对接口的正确使用。该层次的契约无可协商性，如契约不被满足，则系统无法正常执行。

（2）行为契约，通过前置条件和后置条件的断言来限定构件的接口所执行的具体行为。通过这种方式，可以更准确地定义系统执行时所出现的各种错误情况，并明确各构件的职责，以保证各构件都可作为一个原子事务执行。通过契约规定双方的权利与义务，使得该层契约具有了一定的可协商性，各构件可通过断言对前置条件与后置条件进行检验，排除不符合条件的协作者。

（3）同步契约，用于定义各构件间服务调用的全局性行为，目的是通过描述构件所提供服务的依赖性，使构件为客户提供服务的方式具体化。典型的例子是通过并发控制策略来保证构件为多个客户所提供的不同服务不发生相互冲突。

（4）服务质量契约，用于定义服务对象所应具有的服务质量属性，如最长响应时间、平均响应时间、多任务吞吐量、返回结果质量等。该层契约具有最高的可协商性，当有多个功能相似的服务对象时，客户可选择服务质量高的服务对象进行交互。

4.2.2 信任契约描述

对可信性进行约束的契约，应属于第四个层次，即服务质量契约。从社会角度[149]，一种产品的可信性与生产者（或其品牌）的信誉度紧密相关，而这种信誉的累积和提高，在于生产者对于产品质量的承诺及用户对于这种承诺的认可；从生产者角度，产品质量的承诺建立在对产品功能充分测试的基础上；从用户角度，通过实际使用完成对于产品质量的验证和生产者承诺的评估，从而决定对于产品或生产者的可信性程度。上述过程可以总结为一种面向生产者和用户双方的“承诺—评估”机制。这种机制为服务提供者与使用者之间建立信任契约提供了良好的范型。借鉴契约式设计的思想，在网构软件实体之间建立信任契约，通过规定服务提供者与使用者双方的权利和义务来约束网构软件实体之间的关系，以保证其强可信性。信任契约的建立为信任链提供了信任传递过程中的可信认证机制，使得多个实体之间形成了逐级信任认证的关系。

对实体间信任关系进行抽象，可得到信任契约的 3 个基本要素。

1．前置条件

前置条件是对服务请求者进行的约束，规定了使用服务前必须满足的条件，即服务请求者在使用服务前必须满足服务提供者预设的规约，且有义务对自身的规约满足情况进行检验。服务提供者的预设规约包括业务功能规约与可信性规约。业务功能规约是业务过程中必备的信息，包括服务执行所必需的系统参数信息、状态信息等。可信性规约是对服务请求者可信性状态的约束，包括一定的可信性级别、不允许恶意操作等。3.2 节对强可信实体模型的设计中，服务使用者通过可信交互接口

获取服务提供者的可信性规约，并通过自省接口对自身可信性进行监控，以确保满足服务所需的可信性级别，保证了前置条件的可行性。同时，实体的自明性还向服务提供者承诺了自身的可信状态等信息。这样服务提供者也可以对服务使用者的可信性进行验证，更加强化了信任契约的约束力。另外，还可以通过其他实体的历史交互经验来验证某个实体的承诺，以确保验证信息是真实有效的。在此条件下，如果前置条件失败，则表明服务请求者有 bug 存在。

2．后置条件

后置条件是对服务提供者进行的约束，规定了服务执行后必须满足的条件，即服务提供者需要向服务请求者返回正确、有效、可信的结果，如在规定的响应时间内、通过规定接口、以规定格式返回结果。服务提供者通过可信交互接口获取服务请求者的可信性规约，在按照业务要求完成服务后，有义务对此结果进行检验，以确保结果满足规约的要求。目前，在服务提供者验证自身所提供服务的正确性、有效性、可信性方面已有众多研究成果，此处不再赘述。如果后置条件失败，则表明服务提供者有 bug 存在。

3．不变式

前置条件和后置条件作用于一次实体的交互，而不变式则作用于整个实体。在网构软件中，每个实体的行为都受自身目标的驱动，即使对外提供服务也是为了服务于自身目标的实现，而这些目标在实体的生命周期中是始终不变的。不变式的规约保证了实体的任何内部行为、演化都遵从目标的约束，并且在对外部可见的任何时候都必须确保它是一致的，这也保证了实体本身的服务一致性和功能可信性。实体自身不变式的一个外在表现就是 3.2 节中所述由实体自明性属性提供的服务描述，

在任意时刻、任意请求者看来都应该是一致的，而且实体执行其服务或进行自身演化后也不应发生变化，否则会引起服务使用的混乱，对网构软件系统整体的可信性造成不利影响。

上述信任契约的三要素对实体本身及实体间的交互过程进行了约束，确保了服务请求者与提供者双方的利益，达到在契约基础上的可信。

信任契约描述是建立信任契约的基础，服务提供方需要在提供业务服务的同时提供明确的信任契约描述，以便于服务使用者对后置条件的满足情况做出评估。同时服务使用者也会通过信任契约的描述选择自身能满足其前置条件的服务实体进行协商，并建立交互关系，保证了系统在服务交互层面的可信性。通过便于计算机处理的描述语言对信任契约的前置条件、后置条件和不变式这三部分内容进行描述：

```
契约::=<前置条件，后置条件，不变式>
前置条件::=<业务功能规约，可信性规约>
后置条件::=<数据结果，可信性属性>
不变式::=<服务描述>
业务功能规约::=<接口，参数，参数类型>
可信性规约::=<实体自身可信性级别，信任值>
数据结果::=<数据结果值，数据类型>
可信性属性::=<信任值，响应时间>
<xs:element name="Contract">
    <xs:complexType>
        <xs:all>
            <xs:element name="Pre-condition">
                <xs:complexType>
                    <xs:all>
                        <xs:element name="FunctionalRule">
                            <xs:complexType>
                                <xs:attribute name="Interface"/>
                                <xs:attribute name="Parameter"/>
                                <xs:attribute name="Type">
```

```
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="DependabilityRule">
                                <xs:complexType>
                                    <xs:attribute name="DepLevel"/>
                                    <xs:attribute name="TrustWorthness"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:all>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Post-condition">
                    <xs:complexType>
                        <xs:all>
                            <xs:element name="FunctionResult">
                                <xs:complexType>
                                    <xs:attribute name="ReturnValue"/>
                                    <xs:attribute name="Type"/>
                                </xs:complexType>
                            </xs:element>
                            <xs:element name="DependabilityProperty">
                                <xs:complexType>
                                    <xs:attribute name="TrustWorthness"/>
                                    <xs:attribute name="RespondTime"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:all>
                    </xs:complexType>
                </xs:element>
                <xs:element name="Invariant">
                    <xs:complexType>
                        <xs:attribute name=" InvariantDescription "/>
                    </xs:complexType>
                </xs:element>
            </xs:all>
        </xs:complexType>
    </xs:element>
```

信任契约描述机制建立后，各服务请求实体都可通过可信交互接口获得服务提供者的信任契约。服务提供者与服务使用者都有义务根据信任契约对自身所提供的数据及自身可信性情况进行检查，以保证信任的有效传递及系统整体的可信性。

4.3 基于评估的信任衰减过程

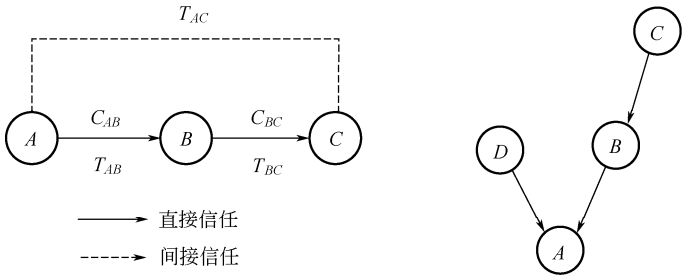
目前网构软件可信性保障的基本思想是用基于黑盒的柔性可信保障技术作为传统基于白盒的刚性保障方法的补充[1]。但这种补充不能割裂可信性两种表述中的内在联系，即实体的可信性属性在达到了某些客观指标后才能满足评估者的需求，从而使评估者产生主观的信任。而且，由于信任值是以评估者的主观感受为依据的，因此，不同评估者所处的环境（如网络环境等）不同，其主观信任值也会受到环境的影响，在计算信任衰减参数时也应考虑此因素。本节按照实体间信任关系传递过程建立贝叶斯网络，以主观推荐信任与客观可信性评估值为基础，提出间接信任传递过程中的信任衰减参数的计算方法。

4.3.1 贝叶斯网络的建立

信任关系通常分为直接信任关系与推荐信任关系。对于直接信任，评估实体在与协作者交互之前，可通过可信交互接口获取其业务功能、可信性保障等信息形式化描述。通过分析与验证这些信息，可得到该实体的功能契合度、接口兼容度、可信性级别等评价指标，以对该实体能

够成功完成协作的概率（即可信性）得出准确的结论。

对于推荐信任，我们认为，信任的衰减不但与评估者对推荐者的信任度有关，还与评估者到目标实体间的通信链路、信息传递安全性等信息传输方面的可信性有关。将实体间信任传递关系抽象成图，如图 4.1（a）所示。实体 B 将实体 C 推荐给实体 A，那么 A 就通过 B 的推荐信息获取对 C 的信任，这样所得到的实体间的信任是间接的，称之为间接信任。



(a) 实体间信任传递关系　　(b) 贝叶斯网络结构

图 4.1　信任传递

图 4.1（a）中，C_{xy} 表示实体 x 与 y 之间的传输可信度，T_{xy} 表示实体 x 对 y 的信任值，x、y 分别表示图中的各实体。

通过建立贝叶斯网络的方法对信任传递过程中的衰减进行计算。在图 4.1（a）中实体 A 通过实体 B 获取实体 C 的信任值时，采用如下步骤建立贝叶斯网络。

（1）选择随机变量{A，B，C，D}，其中 C 代表实体 C 自身的可信性情况，B 代表实体 B 对实体 C 的推荐信任信息，D 代表实体 A 对自身与实体 C 间实际传输可信性情况的评估，A 代表实体 A 对实体 C 的最终信任度。

（2）由于实体 C 的可信性通过实体 B 向实体 A 传播，因此选择变量顺序 α=<C，B，D，A>。

（3）从一个空图 $\mathcal{G}$ 出发，按照顺序 α 逐个将变量加入 $\mathcal{G}$ 中。

（4）最后形成如图 4.1（b）所示的贝叶斯网络结构。

4.3.2 间接信任计算

贝叶斯网络表示一组变量的联合概率分布。图 4.1（b）中显示了变量 A、B、C、D 概率的联合分布。一般地说，贝叶斯网络表示联合概率分布的方法是指定一组条件独立性假设，以及一组局部条件概率集合。联合空间中每个变量在贝叶斯网络中表示为一个节点，需要两种类型的信息。首先，网络弧表示断言“此变量在给定其直接前驱时条件独立于其非后继”。如图 4.1（b）所示，从 C 到 B 存在一条有向路径，称 B 为 C 的后继。其次，对每个变量有一个条件概率表，它描述了该变量在给定其立即前驱时的概率分布。

可以用贝叶斯网络在给定其他变量如 D 观察值时推理出目标变量如 A 的值，也即目标变量的概率分布。它指定了在给予其他变量观察值的条件下，目标变量取每一个可能值的概率，在这里即实体的可信性。在网络中所有其他变量（即实体可信性评估值及其他实体的推荐信任值）都确切知道之后，这一推理步骤是很简单的。

因此，通过贝叶斯网络对网构软件中某实体的间接信任值的计算方法如下，以图 4.1（a）中的实体 C 为例：

$$P(D_C \mid M) = \frac{P(M \mid D_C)P(D_C)}{P(M)} \tag{4.1}$$

（1）计算先验概率：获取其他实体对目标实体的推荐信任值，以此作为先验概率，即式（4.1）中的 $P(D_C)$，这是对实体 C 间接信任值计算的基础依据。

（2）获取观测证据：即式（4.1）中的 $P(M)$，通过评估实体 A 到实体 C 之间的通信链路、信息传递安全性等信息传输方面的可信性值，获取当前系统环境下的观测证据。

（3）计算似然度：通过获取其他实体在与实体 C 交互时的历史记录，建立条件概率表，可计算出在先验概率下系统所处的环境状况，即式（4.1）中的 $P(M \mid D_C)$。

（4）计算后验概率：通过综合其他实体对实体 C 的推荐信任值以及与实体 C 的交互历史，并使用实体 A 目前的观测证据加以修正，可得到实体 A 对实体 C 的最终信任值 $P(D_C \mid M)$。

评估实体对目标实体的信任度取决于评估实体与目标实体间的传输可信度及各推荐实体的推荐信任值，但要通过评估实体对各推荐实体的信任度及推荐实体到目标实体的传输可信度加以修正。如存在多级推荐，每一级的每个实体也按此方法进行综合、汇聚，最终汇聚到评估实体而得到唯一的结果。

4.4　基于网构软件的软件评测支撑平台设计及实验分析

为验证第 3 章、第 4 章所提出模型的合理性，本节结合软件评测中

心的实际应用设计了基于网构软件的软件评测支撑平台，并进行了综合实验。首先根据软件评测中心的业务需求介绍系统平台的设计；然后按照评测服务的业务流程设计实验流程，该流程包含单个实体内部行为、实体间交互行为、信任契约的约束机制、实体间的多级信任推荐等步骤；最后对多级信任推荐实验中的信任衰减参数进行计算，并对实验结果做出分析。

4.4.1 系统平台设计

在软件评测领域中，一方面，软件评测中心拥有软件企业所不具备的多种自动化测试工具及专业化测试人才，但各地软件评测中心的主营业务又有所不同，致使某些软件只能异地评测；另一方面，软件产品日趋复杂，若将软件都发布到软件评测中心的服务器上进行评测，既复杂又难以保证一次运行成功，甚至评测中心无法满足软件发布的硬件需求。因此，基于网构软件平台建立软件评测中心的联盟，通过互联网为用户提供可定制的评测服务，如图 4.2 所示。

各评测中心可提供不同种类的评测服务，如单元测试、压力测试、界面测试等。评测中心之间通过高速网络互连互通、信息共享，通过统一的平台接口向用户提供服务。用户只需要在本地服务器上发布系统，并通过安全连接接入当地软件评测中心的网络，就可以享受到评测联盟所有成员提供的服务。用户可根据自身需要对测试服务进行个性化定制，并支付相应费用。

各软件评测中心的节点都是具有自省性、自明性和自主性的网构软件实体，可根据用户的选择进行测试，如果本地无法满足服务，则自动

向可提供服务的实体申请，由其他实体来提供服务。实体内部处理流程如图 4.3 所示。

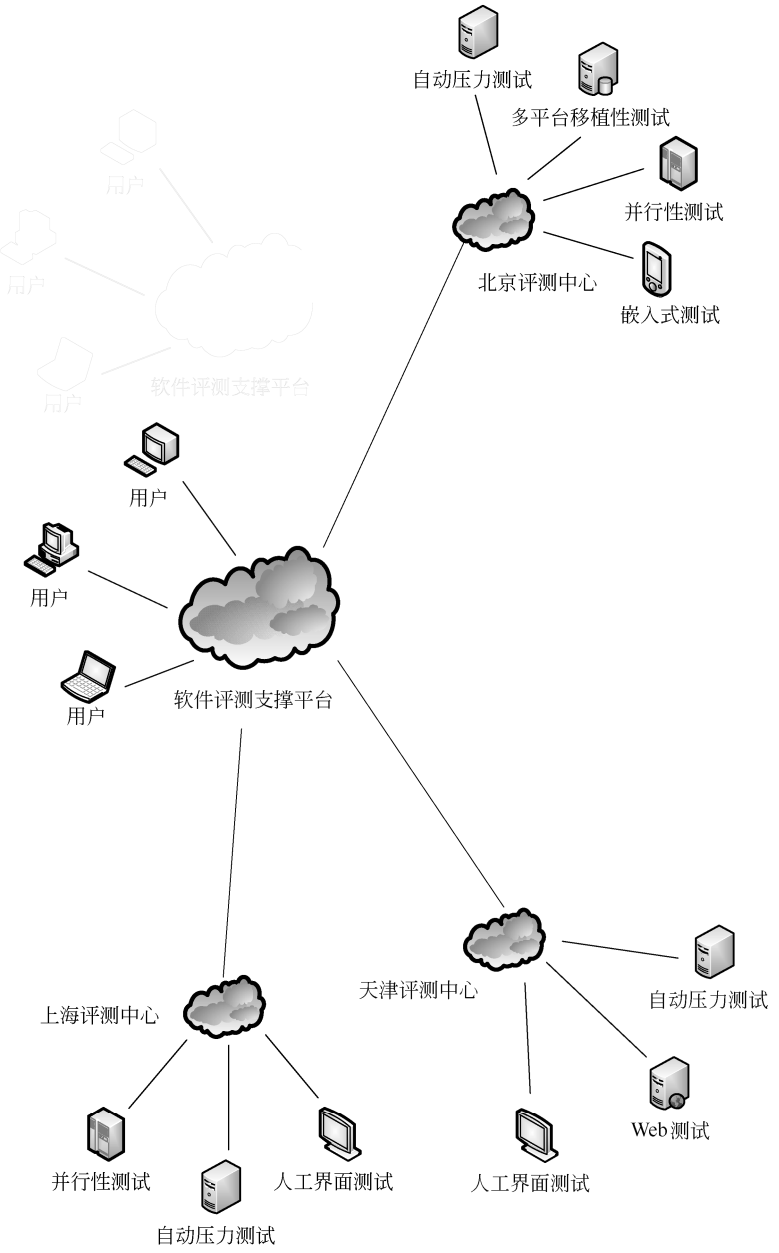


图 4.2　基于网构软件的软件评测支撑平台

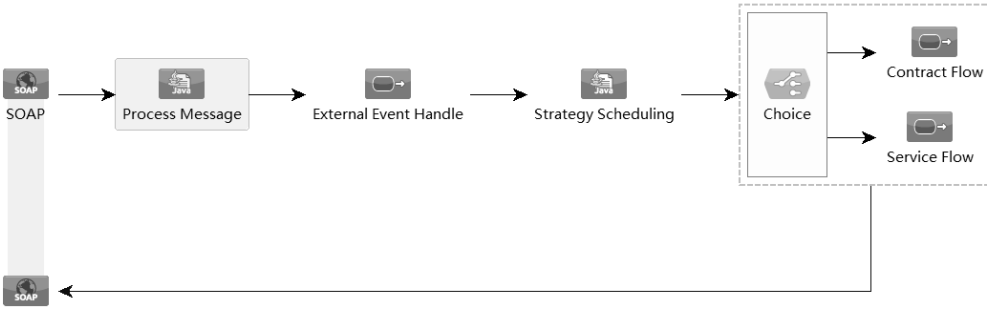


图 4.3　实体内部处理流程图

（1）信息接收：实体通过外部接口接收到与其他实体的交互事件，对事件信息进行解析、处理，并封装成系统内部的信息传输格式。

（2）外部事件处理：通过事件处理总线，向各事件处理构件传递事件信息，由各事件处理构件捕获与自身处理功能相关的事件，如图 4.4 所示。

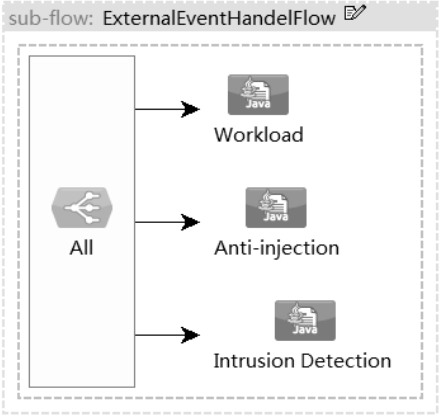


图 4.4　外部事件处理总线

事件处理主要包括流量控制、防注入攻击、入侵检测等功能，以检测可能对实体自身的可信性造成影响的事件。通过事件处理，并结合实

体当前可靠性、可用性、效率等各项可信性指标情况，设置系统可信性级别及各项状态参数。后续的行为都会以此为执行依据，保障了实体实时监控可信性变化情况的自省性。

（3）可信性策略调度：根据系统可信性级别及各指标状态，依据预设的策略激发可信性保障行为，以应对外界环境的变化。例如：监测到有错误注入事件时，对每一次交互都执行相应保障行为，以过滤实体所受到的潜在错误注入攻击。这保证了系统主动应对外界环境变化的自主性。

（4）业务功能处理：业务功能处理分为两种类型，即契约处理和服务处理。当实体识别出外部事件为契约协商请求时，进入契约处理流程，如图 4.5 所示。

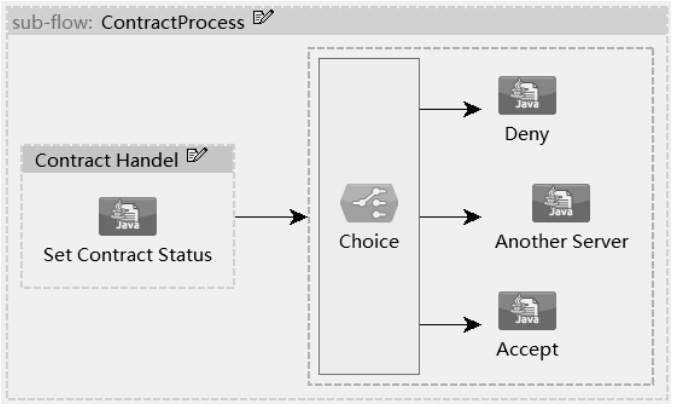


图 4.5　契约处理流程

契约处理构件将契约中携带的协作方信息与本实体的状态、可信性级别、预期目标等参数进行对比，若可接受则通知对方继续协同行为，若无法接受则予以拒绝，若无法提供服务则从自身知识库中查询可提供同种服务的实体并转发服务请求。服务处理则是与通过契约协商后的其

他实体进行具体业务相关的协同行为。

4.4.2 实验设计

依据前述实验场景，按照从个体到整体、先功能后性能的原则，设计实验流程如下。

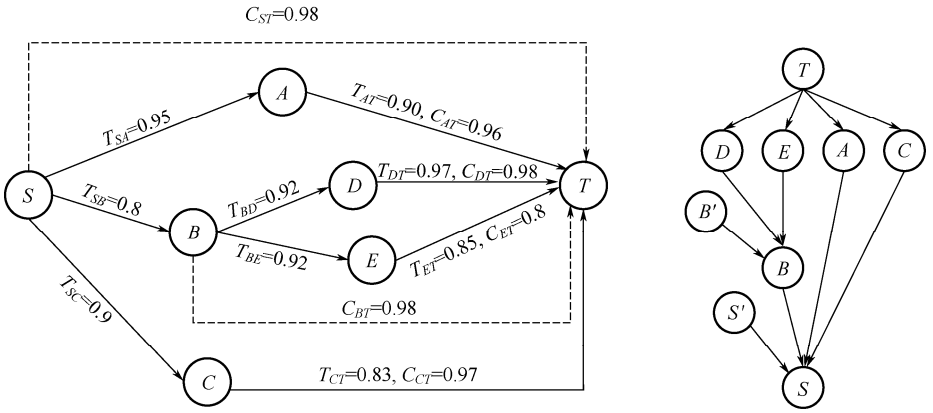
（1）系统初始状态：测试用户向评测中心申请服务，评测中心为其提供正常服务，此时系统处于正常服务状态，可信性保障功能正常。

（2）单个实体内部行为：对于强可信智能实体，在正常的服务请求中加入可能导致服务实体可信性下降的行为，如可能会导致安全性下降的 SQL 语句注入攻击、拒绝服务攻击等。此时，服务实体中应启动适当的可信性保障机制，以应对外界行为的变化。

（3）两个实体间的交互行为：对于基于信任契约的信任约束机制，服务提供实体设定契约的前置条件，如服务请求实体访问服务提供实体的频率应低于每分钟 10 次，客户端应具有对输入参数中非法字符的检测机制等。服务请求实体设定契约的后置条件，如测试方对测试结果的最短响应时间等。某服务请求实体不符合前置条件时，服务提供实体可拒绝服务。服务请求实体对服务提供实体对其契约满足情况进行评估，优先选择与后置条件符合度高的服务实体。

（4）多个实体间的多级信任推荐：基于评估的信任传递，服务请求者与最终服务提供者没有直接交互的历史信息，但可以通过多个中间实体建立间接信任关系。这些实体之间就形成了多层信任传递关系，根据各实体间的信任关系与网络环境状况计算服务请求者对最终服务提供者的信任度。实体间推荐关系如图 4.6（a）所示。

实体 S 需要获取实体 T 提供的服务，可根据实体 A、B、C 的推荐信息计算实体 T 的信任值。而实体 B 综合实体 D、E 对实体 T 的推荐信息后向实体 A 提供信息。图 4.6（a）中所有推荐实体在提供推荐信息的同时，都可通过可信交互接口将自身与目标实体的交互情况发送给评估实体，为信任值的计算提供依据。根据图 4.6（a）中的实体间推荐关系建立贝叶斯网络。



(a) 实体间推荐关系　　(b) 贝叶斯网络结构

图 4.6　多个实体间的多级信任推荐实验场景

① 选择随机变量{S,A,B,C,D,E,T, B', S'}，其中 T 代表实体 T 自身的可信性情况，A,B,C,D,E 分别代表各实体对实体 T 的推荐信任信息，B', S' 分别代表实体 B,S 对自身与实体 T 间实际传输可信性情况的评估，S 代表实体 S 对 T 的最终信任度。

② 由于实体 T 的可信性通过实体 D,E,A,B,C 向实体 S 传播，因此选择变量顺序 α =<T,D,E, B',B,A,C, S',S>。

③ 从一个空图 $\mathcal{G}$ 出发，按照顺序 α 逐个将变量加入 $\mathcal{G}$ 中。

④ 最后形成如图 4.6（b）所示的贝叶斯网络结构。

4.4.3 实验结果分析

实验系统运行在一台配置为 Interl Pentium Dual 1.8GHz 双核 CPU、1GB 内存、Windows XP SP3 操作系统的 PC 上。设置用户仅请求最简单的单元测试功能，以减少业务逻辑复杂度差异给可信性保障性能所带来的误差。实验结果及分析如下。

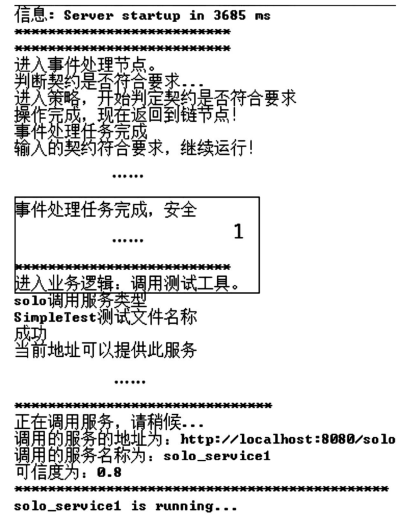
1．单个实体及两实体间交互行为实验

系统实验结果如图 4.7 所示，图中省略了不相关的中间步骤。

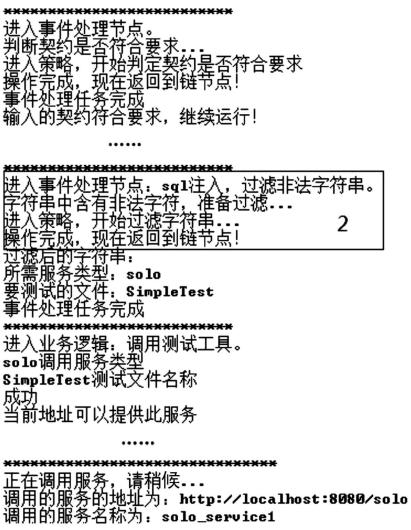
对比图 4.7（a）和图 4.7（b），通过框 1 中的信息可以看出，当服务请求正常时，系统按照规定的业务逻辑进行了服务，并提供了所需结果。框 2 中的信息说明系统发现 SQL 注入攻击行为，此时外部事件处理机制启动，根据既定策略调用相应行为，则外部请求须经过防 SQL 注入攻击处理后再进行业务逻辑处理。同时系统自主修改可信级别，并向外界发送通知。由此可见，强可信智能实体的设计可以保证其自省性、自明性、自主性的实现。

如图 4.7（c）和图 4.7（d）所示，服务请求实体选择可信性高的服务提供实体进行交互，符合网构软件实体间交互的一般机制。服务提供实体设置前置条件访问服务实体最短间隔时间为 2 秒，而对于服务请求实体，由于其需要实时监控服务执行状况，需要每秒访问服务实体，不满足信任契约的前置条件。因此，通过框 3 中的信息可以看出，服务实体首先判断服务请求实体是否满足信任契约前置条件的要求，结果是不满足，则服务实体拒绝提供服务。框 4 中显示了服务请求实体的信息，服务请求被服务提供实体拒绝，服务请求者转向其他服务提供实体。由此可见，通过对信任契约的评估，可以在服务交互前获得实体可信性情

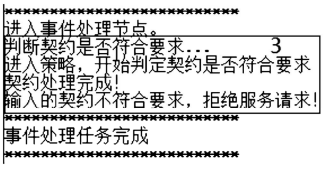
况，同时服务提供实体也拥有了预先对服务请求者进行信任评估的能力。



(a) 系统正常执行



(b) 实体动态行为



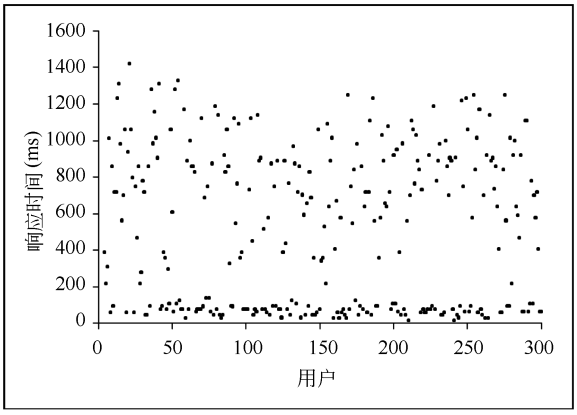
(c) 契约处理服务提供实体

(d) 契约处理服务

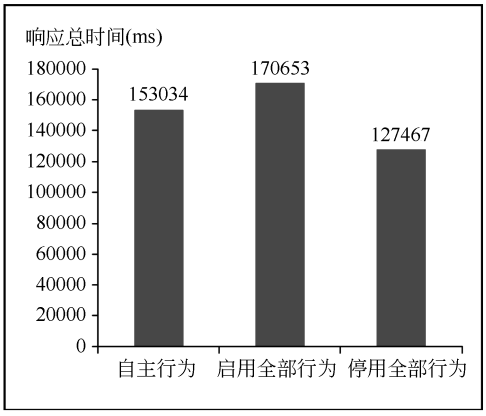
图 4.7　系统实验结果

2．可信性保障性能测试

在自主可信性保障行为、启用全部可信性保障行为、停用全部可信性保障行为三种方式下，分别模拟了 300 个用户的随机测试服务请求。其中契约协商请求与业务服务请求的比例为 3:7，所有请求行为中有危险的攻击行为占 30%。测试结果如图 4.8 所示。



(a) 使用自主行为时请求响应时间



(b) 不同行为方式响应总时间对比

图 4.8　可信性保障性能测试结果

图 4.8（a）是在自主可信性保障行为下各用户请求的响应时间，从

图中可以看出各请求响应时间的分布情况。其中有 30%左右的请求响应时间在 100ms 以下，这是由于这些请求都是单纯的契约协商，并未执行实际的测试业务。请求响应时间在 100～300ms 的数据约占 10%，是由业务请求中的恶意行为被可信性保障机制拦截而造成的。另外契约协商中也有部分恶意行为被拦截。以上数据很好地反映了业务执行过程中各种交互行为的实际情况，表明自主可信性保障行为可以为网构软件实体提供正常的可信性保障。

图 4.8（b）是不同可信性保障行为方式下请求总时间的对比。启用全部可信性保障行为时会对所有请求都执行所有的可信性保障处理操作，而不考虑系统目前的自身及环境状况，因而进行了许多不必要的操作，造成响应总时间最长。启用自主可信性保障行为后，系统会根据当前自身及环境的可信性情况对可信性保障机制进行动态演化，仅执行有针对性的行为，提高了保障效率，减少了请求响应时间。停用全部可信性保障行为后，所有请求都直接进入核心的业务逻辑，虽然请求响应时间有所减少，但可信性方面没有任何保障。而且所减少的响应时间中有大部分是系统出错返回而没进行后续处理造成的，这种情况对实际应用中可信性的不利影响是显而易见的。另外，启用自主可信性保障行为后所增加的响应时间并不多，但可以有效保障系统的可信性，充分说明我们提出的可信性保障机制是合理有效的。

3．多级信任推荐中信任值的计算

按照逐级综合计算的方法，先综合实体 D、E 对 T 的信任值来计算实体 B 对 T 的信任值。

首先，获取各实体对目标实体的信任值，从图 4.6（a）中可以得出，即 T_{DT}=0.97，T_{ET}=0.85。

然后，根据实体 D、E 与 T 的交互情况建立如表 4.2 所示的条件概率表。

表 4.2　实体交互情况条件概率表

通信链路质量	信息传递安全性	网络时延	交互结果
1	1	1	1
1	1	1	1
0	0	0	0
1	1	1	0
1	1	1	1
0	1	1	0
1	0	0	0

表 4.2 中显示了对传输可信性相关属性的不同满足情况所带来的交互结果差异。表中值为 1 的项表示满足该情况，值为 0 的项表示不满足。例如，第一条数据表示通信链路质量、信息传递安全性、网络时延都满足标准的情况下交互成功。最后一条数据表示仅有通信链路质量满足标准的情况下交互失败。由此可计算出各推荐实体与目标实体的传输可信性值，即不同传输条件下交互的概率。在图 4.6（a）中表示为 C_{DT}=0.98，C_{ET}=0.8。表 4.2 由汇总各推荐实体的交互历史而得，内容没有全部列出，仅列出几条代表性数据。

最后，结合实体 B 自身的传输可信性观测证据，即 $C_{BT}=0.98$，计算出实体 B 对 T 的信任值：

$$T_{BT}=P(D_T\mid M_B)=\frac{P(M_{DE}\mid D_T)P(D_T)}{P(M_B)}=0.98 \tag{4.2}$$

得到实体 B 对 T 的信任值后，综合实体 A、B、C 对 T 的信任值来

计算实体 S 对 T 的信任值：

$$T_{ST} = P(D_T \mid M_S) = \frac{P(M_{ABC} \mid D_T)P(D_T)}{P(M_S)} = 0.9 \tag{4.3}$$

结果分析 1：从上述计算过程可以看出，间接信任的计算可以通过推荐信任的逐级综合而来。在计算当前级别的信任度前，首先计算前一级别实体对目标实体的信任度。所有间接信任度都是通过推荐信任度与传输可信性计算得来的，所有实体在计算间接信任前都需要通过评估获取自身到目标实体的传输可信性，反映了主观推荐与客观评估相结合的特点。

结果分析 2：由 T_{ST} 的计算过程可以看出，在实体 S 到 T 的网络环境与 A 到 T、B 到 T、C 到 T 相差不大的情况下，实体 S 对信任度最高的实体 A 的推荐信任采信度最高，即 T_{ST} 与 T_{AT} 最接近。这反映了间接信任是在对推荐实体信任度的基础上由推荐信息综合而来的。

结果分析 3：由 T_{BT} 的计算过程可以看出，在实体 B 对实体 D、E 的信任度相同的情况下，虽然实体 E 所提供关于实体 T 的推荐信任值较低，但由于实体 E 到 T 的网络环境较差，由网络环境造成服务失败的可能性较大，因此采信度不高。而实体 B 到 T 的网络环境与 D 到 T 最相近，因而对 D 的推荐信息采信度最高，即 T_{BT} 与 T_{DT} 最接近。这反映了传输可信度对推荐信任的修正作用。

结果分析 4：T_{BT} 的计算结果比所有推荐者的推荐信任值 T_{DT}、T_{ET} 都高，体现了传输可信性对信任传递衰减参数的反向修正作用。推荐实体的网络环境越差，则造成目标实体服务失败的概率越高，因此当推荐实体与评估实体的网络环境状况有差别时，特别是评估实体的网络环境较好时，须通过修正参数提高目标实体在当前环境下的可信度，以反映评估实体的真实网络状况。

第 5 章

基于分层 Petri 网的网构软件可信性演化模型

网构软件实体间的信任关系需要经历契约协商与信任演化两个过程。契约协商是协同双方建立互信关系的过程，即网构软件实体选择可信性高的实体进行协同并只允许可信性高的实体访问其服务。该过程反映了实体内部机制的运行与变化。信任关系建立后，由于网构软件实体能够感知外部网络环境的动态变化，并随着这种变化对自身行为进行静态的调整和动态的演化，以使系统具有尽可能高的用户满意度。而且，由于用户需求的多样化和个性化以及投资回收等因素，一个软件系统往往存在时间较长，因此，网构软件的演化过程也在长时间内持续不断，实体间的信任关系也会随之演化。这就需要有一个系统级的统一模型对信任的协商与演化过程进行持续管理，并同时对实体内部与实体之间运行机制的变化情况进行描述。本章针对此需求，在分析网构软件体系结构的基础上，建立系统结构模型，并根据此模型建立基于分层 Petri 网的可信性演化模型，从系统整体组成结构与各实体内部契约协商策略两方面对网构软件演化过程中的可信性演化情况进行描述与管理，为网

构软件及其实体的设计、开发、部署与演化提供技术保障与支持。

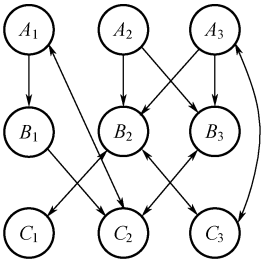
5.1　网构软件结构分析

同构件软件相比，组成网构软件的基本元素不是构件或模块，而是软件实体，它是由软件开发商或普通用户提供的能满足某个网构软件所需服务的实体，而且各实体之间可以自主实现协同性。各实体均为散布在互联网中的软件系统，彼此独立、自主运行，不受任何机构或组织的统一控制，在开放、动态和多变的环境中进行协同。网构软件既服务于处在不同时区的用户，也作为一个软件实体服务于随时到达的其他网构软件的请求。上述体系决定了网构软件的结构是非常复杂的，因此，为评估网构软件系统的可信性，须对其组成结构进行清晰良好的分析。我们根据网构软件的特性，定义了其结构相依性和语义相依性。

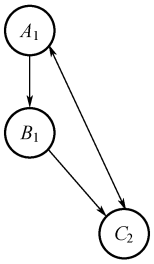
定义 5.1（结构相依性） 在组成网构软件的实体集合中，根据实体所实现的功能的不同，可以将其分成不同的子集合，相同或相似功能的实体成为一个子集合；各子集合中的某两个或多个实体在使用中会有一定互相依赖的关系，定义为结构相依性。

例如，假设网构软件系统的实体集合根据功能的不同可以分成三个子集合，子集合 A 有实体 A_1，A_2，A_3；子集合 B 有实体 B_1，B_2，B_3；子集合 C 为基础实体层，有实体 C_1，C_2，C_3。A_1 在结构上依赖 B_1 和 C_2，而 C_2 在完成功能的数据流动中，又需要动态绑定到 A_1，等等。由此可得到系统的结构相依性图，如图 5.1（a）所示。为了后续的处理方便，将其转化为三棵结构相依性树，分别以子集合 A 的实体 A_1，A_2，

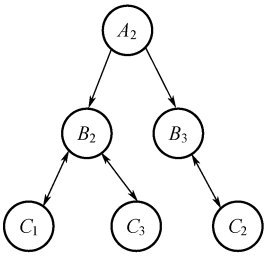
A_3 为根节点，如图 5.1（b）～（d）所示。



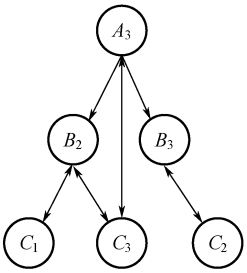
(a) 相依性图



(b) 相依性树1



(c) 相依性树2



(d) 相依性树3

图 5.1　结构相依性图与结构相依性树

定义 5.2（语义相依性） 在同一实体子集合中，各实体为完成应用系统的特定功能而采取的多实体之间的交互顺序，即为这些实体之间的语义相依性。某些实体子集合中，由于实体功能的独立性，也可以不存在语义相依性。

对于每一实体层内部的语义相依性，由于实体之间有顺序关系，所以用矩阵来表示，如果本实体层有 N 个实体，则矩阵为 $N\times N$。

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad a_{ij}=（0,1）$$

当实体层 A 中的 A_i 必须在 A_j 之前使用时，$a_{ij}=1$；否则，$a_{ij}=0$。

5.2　Petri 网用于网构软件可信性演化的相关研究

5.2.1　Petri 网基本概念及性质

Petri 网[150]是一种图形化表示的组合模型，以研究模型的组织结构和动态行为为目标，着眼于系统中可能发生的各种状态变化及变化之间的关系；具有表达直观、易懂易用、数学定义严格的优点；既可以用于静态结构分析，又可以用于动态行为分析；与传统的逻辑方法相比，更能够清晰地表示出系统中的并发行为。

Petri 网的形式化描述如下。

满足下列条件的三元组 $N=(P,T;F)$ 称为一个网：

（1） $P\cap T=\propto$（二元性）；

（2） $P\cup T\neq\propto$（网非空）；

（3） $F\subseteq((P\times T)\cup(T\times P))$；

（4） $\mathrm{dom}(F)\cup\mathrm{cod}(F)=P\cup T$。

其中： $\mathrm{dom}(F)=\{x\mid\exists y:(x,y)\in F\}$

$\mathrm{cod}(F)=\{y\mid\exists x:(x,y)\in F\}$

P 和 T 分别称为 N 的库所集和变迁集，F 为流关系；库所集和变迁集是有向网的基本成分，流关系是从它们构造出来的；库所和变迁是两类不同的元素，所以 $P\cap T=\propto$，而 $P\cup T\neq\propto$ 表示网中至少有一个元素；每一个库所表示一种资源，变迁是资源的流动，由流关系规定，所以变迁只能与库所有直接关系； $\mathrm{dom}(F)\cup\mathrm{cod}(F)=P\cup T$ 表示不存在不参加任何变迁的资源和不引起资源流动的变迁。

设 $N=(P,T;F)$ 为一个网，对于 $x\in P\cup T$，记

$$\dot{}x=\{y\mid y\in P\cup T\wedge(y,x)\in F\}$$

$$x^{\dot{}}=\{y\mid y\in P\cup T\wedge(x,y)\in F\}$$

称 $\dot{}x$ 为 x 的前置元素集或输入集， $x^{\dot{}}$ 为 x 的后置元素集或输出集。

一个网结构往往对应于实际的某一系统结构，而系统的运行会带来不同的系统状态，故而考虑为网结构添加标识来反映网的状态的转变。对于一个标识网[150]，对其中的变迁元素，应该给出变迁元素的发生规则才能够使 Petri 网模型模拟系统动态行为。

一个 Petri 网系统是一个标识网 $\Sigma=(N,M)=(P,T;F,M)$，并有如下的变迁使能条件和发生规则：

（1）对 $t\in T$，若 $\forall p\in\dot{}t:M(p)\geqslant 1$，称 t 在标识 M 使能，记作 MIt>。

（2）若 t 在 M 下使能，则 t 在 M 下可以发生，从 M 发生 t 得到新的标识 M'（记为 $M[t>M'$），对 $\forall p\in P$：

$$M'(p)=\begin{cases}M(p)-1, & \text{当}\quad p\in {}^{\bullet}t-t^{\bullet}\text{时}\\ M(p)+1, & \text{当}\quad p\in t^{\bullet}-{}^{\bullet}t\text{时}\\ M(p), & \text{其他}\end{cases}$$

网系统中变迁的发生改变了网系统的标识（状态），在新的标识下如果有变迁使能则可以继续发生下去，形成整个 Petri 网系统的运行，可以对实际系统的动态行为进行模拟。

设 Petri 网系统 $\Sigma=(P,T;F,M_0)$，若 $\exists M_1,M_2,\cdots,M_k\in R(M_0)$，使得 $\forall 1\leqslant r\leqslant k,\exists t_r\in T:M_r[t_r>M_{r+1}$，则称变迁序列 $\sigma=t_1t_2\cdots t_k$ 在 M_1 下使能，M_{k+1} 从 M_1 是可达的，记为 $M_1[\sigma>M_{k+1}$。

如果一个 Petri 网模型确切地描述了一个系统的结构和运行，那么这个系统所具有的一些性质也会在其 Petri 网模型上得到体现。Petri 网模型的行为性质主要包括有界性（Boundedness）、活性（Liveness）、可达性（Reachability）等[151]。

可达性：对于 Petri 网系统 $\Sigma=(P,T;F,M_0)$，如果 $\exists\sigma\in T^{*}$，$M_0[\sigma>M_1$，则称标识 M_1 是从 M_0 可达的，$R(M_0)$ 为 M_0 的所有可能可达标识集。

可达标识集是 Petri 网任何可能发生序列所能进入的全部状态的集合，通过此集合可以标识出系统所有可能的状态，为许多应用问题提供求解依据。

有界性：称 Petri 网系统 $\Sigma=(P,T;F,M_0)$ 是有界的当且仅当 $\forall M\in R(M_0),\forall p\in P,\exists k\geqslant 0$ 使得 $M(p)\leqslant k$。

安全性：称 Petri 网系统 $\Sigma=(P,T;F,M_0)$ 是安全的当且仅当 $\forall M\in R(M_0),\forall p\in P,M(p)\leqslant 1$。

安全性对应于 Petri 网系统运行过程中的状态，有界性用来刻画 Petri 网系统描述的系统是否存在溢出，而安全性则一般对应于实际系统运行时的一些安全性约束。例如，当某一库所描述某一操作时，该库所的安全性能够保证不会重复启动某一正在执行的操作。

活性：称 Petri 网系统 $\Sigma=(P,T;F,M_0)$ 是活的当且仅当 $\forall t\in T,\forall M\in R(M_0),\exists M_1\in R(M)$，使得 $M_1[t>$。

Petri 网系统的活性对应于实际系统能否顺利运行，系统运行中可能出现的一个死锁对应 Petri 网系统的一个死标识，因此可以通过对死标识的形成、结构等方面的分析对系统死锁进行等价分析。

Petri 网目前已在并行计算[152]、服务组合[153]、工作流建模[154]、柔性制造[155]、分布式数据库[156]等方面得到广泛应用。由于 Petri 网能对异步、并发的计算机系统进行良好描述，因此可以用于网构软件可信性演化的建模。本节着重分析与使用 Petri 网对网构软件实体内部机制及实体间信任关系演化建模相关的研究。

5.2.2 基于 Petri 网的开放环境下软件可靠性评估方法

文献[157]针对基于体系结构的可靠性评估方法难以处理互联网环境下带并行结构的系统的问题，提出了一种开放环境下的软件可靠性评估方法。该方法定义可靠性为对整个系统成功执行的期望，并且系统成功执行时其所有组件也全部按用户的期望执行。组件的重要性定义为该组件可靠性变化时对系统产生影响的比率。系统可靠性：

$$R(\Omega)=E(\prod_{i=1}^{n}R(C_i)^{t_i})$$

其中，Ω代表系统，$C_1,C_2,\cdots,C_n$为系统的各组件，$t_1,t_2,\cdots,t_n$为系统一次运行过程中各组件的运行次数。

组件 A 对系统的重要性：

$$D(\Omega,A)=\frac{\partial R(\Omega)}{\partial R(A)}$$

借鉴工作流方面的工作使用 Petri 网对软件体系结构进行建模，使用变迁描述软件的各个组件，将复杂系统分解为 6 种典型结构（图 5.2），并给出各结构可靠性的计算方法，如表 5.1 所示。

表 5.1　系统结构及其可靠性计算方法

结构	描述	可靠性计算方法
顺序	各个组件依次执行，如图 5.2（a）所示	$\begin{cases}R(S)=\prod_{i=1}^{n}R(C_i)\\D(S,C_i)=\sum_{j=1}^{n}[D(C_j,C_i)\cdot\prod_{k=1,k\neq j}^{n}R(C_k)]\end{cases}$
分支	程序将从几条候选路径中选出一条来执行，如图 5.2（b）所示，图中数字表示各分支的概率	$\begin{cases}R(S)=\sum_{i=1}^{n}[R(C_i)\cdot P(C_i)]\\D(S,C_i)=\sum_{j=1}^{n}[D(C_j,C_i)\cdot P(C_j)]\\P(C_i)\text{为各组件执行概率},\sum_{i=1}^{n}[P(C_i)]=1\end{cases}$
循环	表示某个组件将被重复执行，如图 5.2（c）所示，图中数字表示各分支的概率	$\begin{cases}R(S)=R^n(A)\\D(S,A)=n\cdot R^{n-1}(A)\end{cases}$
并行	表示几个组件都会被执行，如图 5.2（d）所示	$\begin{cases}R(S)=1-\prod_{i=1}^{n}[1-R(C_i)]\\D(S,C_i)=\sum_{j=1}^{n}D(C_j,C_i)\cdot\prod_{k=1,k\neq j}^{n}[1-R(C_k)]\end{cases}$
调用	组件间相互交互的一种主要手段，看做顺序执行	同顺序结构
Goto	用于表示递归、异常等不易表示的复杂结构，将跳转的组件之间用一个库所连接起来	同顺序结构

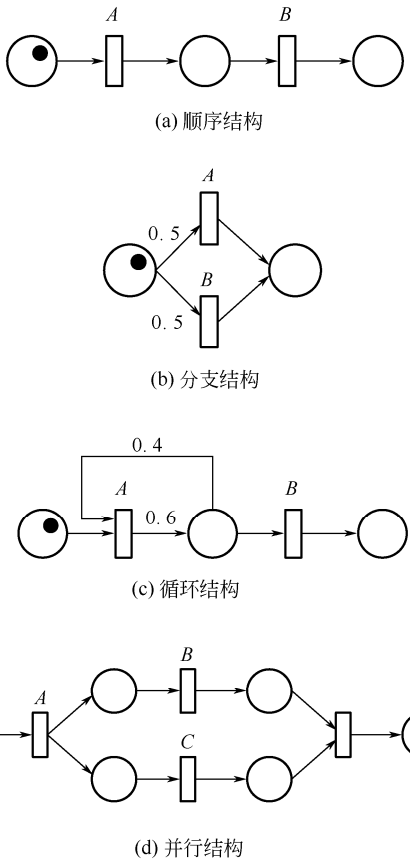


图 5.2　典型结构的 Petri 网描述

在计算系统可靠性时，先将系统分解为一系列进程，即不含并行结构的单元；然后按照前述各结构的计算方法计算出各进程的可靠性；最后将进程的可靠性组合成系统整体的可靠性。

5.2.3　基于 Petri 网自动信任协商模型

对于实体间协同的信任机制问题，通过对参与实体的授权和身份验证确保系统运行的安全性是可信研究的基础问题之一；然而，随着软件

应用向开放、跨组织和管理域的方向发展，如何在不可确知系统边界的前提下实现有效的身份认证和权限控制，如何对跨组织和管理域的协同提供身份可信的保障成为新的问题[61]。自动信任协商为此问题提供了较好的解决途径。

自动信任协商[158]是指通过逐步向对方暴露数字证书，在陌生者之间建立信任关系的一种访问控制方法。当访问者与资源/服务提供方不在同一个安全域时，常规的访问控制方法不能有效地对访问者的行为进行控制，自动信任协商则可以为合法用户访问资源提供安全保障，防止非法用户的非授权访问。

Winsborough 等人[158]提出了自动信任协商（Automated Trust Negotiation，ATN）的概念，并成为当前的一个重要研究方向；BYU 大学 ISRL 实验室的 Seamons 和 UIUC 大学 Winslett 等人联合承担了 ATN 的研究项目 TrustBuilder，他们开展了大量的研究工作，奠定了扎实的应用基础[159]；Stanford 大学的“动态协作的快捷管理”项目和 NAI 实验室的“基于属性访问控制”项目也在开展有关 ATN 的理论和应用研究[160]。在国内，北京航空航天大学怀进鹏等研究的 COTN 系统[161]引入了基于契约的信任协商方法，既在契约确立过程中预先终止了无法进行的协商请求，又在契约约定下的协商过程中研究了对信任证和访问控制策略中隐私信息的保护，能够高效、可靠地自动建立信任关系；中科院软件所的冯登国等提出的吝啬语义信任协商框架[162]，使交易双方的协商引擎可以快速有效地从由身份断言权威签发的身份断言证书中，自动计算出批露最少信息且符合对方策略的身份断言集进行出示，避免了完整属性证书的直接出示，降低了信任建立过程中身份信息被披露的程度。

文献[163]将 Petri 网概念引入信任协商策略中，提出基于 Petri 网的

信任协商模型 Negotiation Petri Net。

定义 Negotiation Petri Net 为一个三元组，$NP=\{P, T; F\}$，其中 $P=\{C_i$, C_i 是协商双方的信任证$\}$，$T=\{t_i, t_i$ 是 P 中除对未被保护信任证的其他信任证的运算关系$\}$，$F=\{$与信任协商机制对应的 T 和 P 之间的流关系$\}$。

使用从变迁节点到库所节点的流关系表示 $\wedge$ 运算，从库所节点到变迁节点的流关系表示 $\vee$ 运算，将令牌放置在与未被保护信任证相对应的库所中，即 TRUE 信任证。这样就建立起了对于自动信任协商中访问控制机制的 Petri 网表示。建立 Petri 网的目的是找到一个信任协商中的安全披露序列，通过寻找原 Petri 网的反序网（Reverse Negotiation Petri Net）来完成。

定义三元组 $NP=\{P,T;F^{-1}\}$ 为 $NP=\{P,T;F\}$ 的 Reverse Negotiation Petri Net，其中 $F^{-1}=\{(x,y)|(y,x)\in F\}$。

设最初请求的服务 S 信任证对应的库所为根节点，则安全披露序列为满足如下条件的解路径中所有信任证的反序。

（1）包含根节点 S。

如果节点 $n\in P$，则路径中包含集合 $\{t_i|(n,t_i)\in F\}$ 中的某一变迁节点；

如果节点 $n\in T$，则路径中包含集合 $\{C_i,S_i|(n,C_i)\in F,(n,S_i)\in F\}$ 中的所有库所节点。

（2）路径以某一带有令牌的库所节点终止。

5.3 网构软件系统建模

对网构软件来说，其可信性保障不仅是流程的处理，更重要的是实

体间的信任关系。因此，软件系统的建模应着重面向对实体间相互关系的描述，并兼顾其演化。同时，网构软件间的交互也不等同于自动信任协商，还涉及网构软件实体的多态性，即实体可以为不同用户提供个性化的服务，因此需要根据不同可信性级别建立多层次的访问控制策略。另外，还需要将这两方面联系起来，以便于从系统整体的角度进行统一管理与描述。

分层 Petri 网是在经典 Petri 网基础上逐步发展起来的，融合了经典 Petri 网所具有的优点，又具有面向对象的性质，便于自顶向下设计，对于复杂系统的建模清晰明了，有助于用户理解并对系统分析和规约。而网构软件是由分布于互联网上的多个有自主性的软件实体组成的，其可信性分析既要考虑实体内部的访问控制机制，又要考虑实体间的相互协作，且实体间既有静态的相依性，又有动态的交互关系。因此，适合采用分层 Petri 网进行建模与分析。

本节在结构分析的基础上，使用分层 Petri 网建立网构软件系统模型。通过上层网描述网构软件实体间的静态结构与动态协同关系，通过下层网描述各实体内部的访问控制机制。这样，在涉及实体间初次建立信任关系时，以及交互过程中对于特定实体内部的信任管理机制可以通过下层 Petri 网进行具体分析。而对网构软件系统整体进行可信性分析时，软件实体间相互协作与演化带来的改变可以在上层 Petri 网中直观地反映出来，并且不会受到具体细节影响。

5.3.1　系统结构建模

网构软件系统是由实体及其连接子组合而成的，基于 5.1 节所述的

网构软件结构模式，从相依性矩阵中选取元素 $a_{i,j}$=1 的实体，建立基于 Petri 网的网构软件实体组成结构模型。

定义 5.3（分层实体网）分层实体网是网构软件系统结构基于分层 Petri 网的反映，它可以形式化定义为 $N_{\mathrm{HEN}}=(P,T,W,i,o,M)$，其中：

（1）P 是库所的有限集合，代表组成相依性图的所有网构软件实体；

（2）T 是变迁的有限集合，代表客户实体可以与服务实体建立起可信的交互关系。此处的变迁均为抽象变迁，内部行为可由下层 Petri 网进行细化；

（3）$W\subseteq(P\times T)\cup(T\times P)$，为连接库所和变迁的有向弧，用于描述库所和变迁之间的相互控制依赖关系，方向由上层实体指向基础实体；

（4）i,o 分别为输入库所和输出库所，满足条件：

$$\dot{i}=\{x\in T\mid(x,i)\in W\}=\varnothing$$

$$o^{\cdot}=\{x\in T\mid(o,x)\in W\}=\varnothing$$

（5）$M:P\to\{0,1,2\cdots\}$ 为状态标识，当库所拥有托肯时，表示该实体拥有执行权，需要调用服务，但当且仅当细化变迁的下层 Petri 网运行成功时，该变迁才能发生。

根据上述定义，结合实体之间的相依性可得出实体间基本协同关系的 Petri 网表示。

1．顺序关系

这是网构软件实体协同的基本关系，当一个实体使用另一个实体的服务时，即形成了此类关系，如图 5.3 所示。

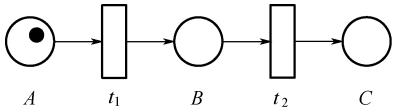


图 5.3　顺序关系

实体 A 具有服务需求，其运行依赖于实体 B，此时实体 A 需要调用实体 B 的服务才能够正常执行，而调用能否成功，取决于变迁 t_1 能否正常运行。变迁 t_1 运行后，实体 B 具有服务需求，变迁 t_2 可依此顺序执行。

2．多对一与关系

当多个实体需要组合以完成某功能，且这些实体都需要第三方提供支持时，形成此类关系，如图 5.4 所示。

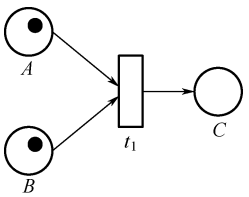


图 5.4　多对一与关系

实体 A 和 B 都需要实体 C 的服务，且 C 必须同时得到 A 和 B 的输入参数才能够执行。C 分别对 A 和 B 的服务请求进行可信性验证，即变迁 t_1 的下层 Petri 网需要运行成功。成功执行后 A 和 B 的托肯被消耗掉，C 产生一个新的托肯。

3．一对多与关系

当一个实体的执行需要其他多个实体提供服务时，客户实体与服务实体间形成此类关系，如图 5.5 所示。

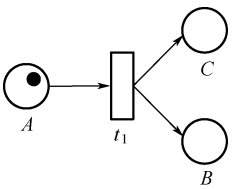


图 5.5　一对多与关系

实体 *A* 依赖于实体 *B* 和 *C* 的服务，必须同时得到 *B* 和 *C* 的服务才能够执行。此时，*B* 和 *C* 分别对 *A* 的服务请求进行可信性验证，成功执行后 *A* 的托肯被消耗掉，*B* 和 *C* 各产生一个新的托肯。此时，变迁 t_1 的下层 Petri 网被设置为可信性要求最高的服务访问控制策略。

4．多对一或关系

当有多条路径可以完成某功能时，任意路径的成功协同都可满足用户的需求，这样就形成了此类关系。如图 5.6 所示。

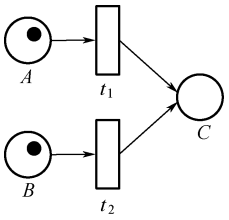


图 5.6　多对一或关系

实体 *A* 和 *B* 都需要实体 *C* 的服务，且 *C* 得到 *A* 和 *B* 中任意实体的输入参数即可执行。此时，*C* 分别对 *A* 和 *B* 的服务请求进行可信性验证，即运行变迁 t_1 和 t_2 的下层 Petri 网。成功执行后消耗对应库所中的托肯，C 产生一个新的托肯。未执行成功的变迁则丧失发生权，即未通过可信性验证。

5．一对多或关系

当一个实体的执行有多个备选实体时，与任意实体协同成功都可满足用户需求，则形成此类关系，如图 5.7 所示。

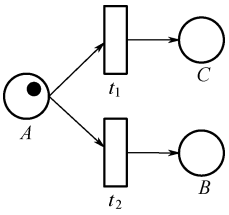


图 5.7　一对多或关系

实体 *A* 依赖于实体 *B* 或 *C* 的服务，得到其一即可执行。此时，*B* 和 *C* 分别对 *A* 的服务请求进行可信性验证，任何一个变迁成功执行后 *A* 的托肯都会被消耗掉，成功执行的变迁所对应的库所中产生一个新的托肯。未执行成功的变迁则丧失发生权，即不再需要提供服务。

5.3.2　契约协商策略建模

网构软件服务实体须向不同客户实体提供个性化的服务，同时要监控可信性的演化情况，以便随时采取自主行为。因此，在实体内部需要建立层次式、动态性服务访问控制策略。而信任建立的过程就是客户实体按照服务访问控制策略逐步提供自身可信性信息的过程。我们引入自动信任协商中的若干概念以支持访问控制策略的设计，并针对网构软件可信性演化的要求，对其中部分概念进行进一步限定。

1．信任证

属性信任证[164]是一种经权威中心（信任证颁发机构）签名的数字断言，且具有可存储性和可验证性；信任证中包括所有方的属性信息，其优势在于能够为陌生方进行授权，从而无须为信任证建立统一的管理机构。在网构软件中，信任证可以是由权威机构对实体可信属性认证的数字证书。可信属性可以是实体拥有者具有的某种身份（如某机构的成员）或实体可信性某些子属性的评估结果（如可靠性、安全性等），还可以是实体本身所具有的某些能够提高系统整体可信性的性质（如平均响应时间、安全无病毒等）。如果客户实体具有服务实体所需的某些数

字证书，则具有相应的可信性值。当可信性值积累超过一定阈值时，客户实体就达到了某个信任等级。

2．访问控制策略

访问控制策略[165]用来保护资源不被合法用户非授权访问，从而规范合法用户对资源的操作。基于 Petri 网的层次式服务访问控制策略定义如下。

定义 5.4（访问控制网）定义一个访问控制 Petri 网表示为一个五元组，$N_{\mathrm{ACP}}=(P,T,W,i,o,M)$，其中：

（1）$P=\{P_S,P_C\}$ 是库所的有限集合，其中，P_S 代表资源，P_C 代表证书。受限资源和证书用不带托肯的库所表示，公开的可直接访问的资源和证书被认为是授权的，用带托肯的库所表示。

（2）T 是变迁的有限集合，代表信任协商过程中对资源或证书的访问约束条件。每个变迁都含有一个权值，用以表示该变迁触发所需的信任值。每个变迁的输出库所都对应一个由访问该约束条件控制的资源与证书集合。

（3）$W\subseteq(P\times T)\cup(T\times P)$，为连接库所和变迁的有向弧，用于描述库所和变迁之间的相互控制依赖关系。库所指向变迁的弧表示与关系，变迁指向库所的弧表示或关系。

（4）i,o 分别为输入库所和输出库所，满足条件：

$$\cdot i=\{x\in T\mid(x,i)\in W\}=\infty$$

$$o^{\cdot}=\{x\in T\mid(o,x)\in W\}=\infty$$

（5）$M:P\to\{0,1\}$ 是状态标识，每个库所最多只包含一个托肯，并且在满足触发条件时从前一个库所转移到后一个库所中，表示资源或

证书满足访问控制策略后转入下一步。每个库所中的托肯包含一个权值，用以表示该证书的信任值。当用户提供的证书信任值之和达到变迁的信任值时，变迁才会执行。此时消耗掉该变迁所有输入库所的托肯，在输出库所中产生一个托肯，且该托肯的信任值等于该变迁的信任值。

由于网构软件可以为不同角色的用户提供个性化的服务，访问控制策略中不仅像自动信任协商那样会涉及对信任证的访问，还会涉及对服务资源的使用。因此设计层次递进式的访问控制策略，当用户所提供的信任证达到某阈值时可使用相应服务，若在此基础上再提供特定的信认证，则可继续访问下一层次服务。这样既保证了服务的个性化、安全性，也可使用户不致一次披露过多的信认证。同时，服务实体也可对各信任证颁发机构设置信任值，信任值越高，访问服务所要提供的信任证越少。

3．协商策略

协商策略[166]是协商双方在建立信任关系中所采取的暴露证书和访问控制策略的方式。服务实体根据客户实体要访问的资源所处的信任等级，按照访问控制策略路径生成信任证披露序列，每次披露该路径上相关的信任证，当经过任一路径达到服务资源对应的变迁并且满足信任度阈值时可访问资源。在此过程中，用户也可对服务实体对信任契约中后置条件的满足情况（即服务实体的信认证）进行检验，以决定是否继续后续请求。若用户想要请求下一层次服务，则在此基础上继续提供相应的信任证。若客户实体不具有路径上的某个信任证，则系统可引导客户通过其他可选路径达成。

5.4 网构软件系统运行中可信性演化机制

由于网构软件的动态性、演化性，对网构软件实体可信性的评估，不能仅依赖于客户实体在当前协作活动中所提供的信息，还应对客户实体进行交互活动的历史信息加以使用。通过历史交互信息对系统当前执行过程进行反馈，以修正系统当前可信性的结果。同时可根据实体的实时交互情况判断其安全性是否下降，并对访问控制策略进行调整。当客户实体的可信性不足以达到某个信任等级时，则无法使用该等级的服务，实体需要重新建立连接并再次出示符合要求的信任证以获得服务实体的信任。具体演化机制如下。

1．系统初始化

按照结构相依性图，由最顶层实体开始，设定所有需要建立连接的服务实体。按照 5.3 节所述结构定义逐级建立上层 Petri 网中的各组成元素。其中所有库所的结构和连接关系可由相依性图确定，所有变迁均设为抽象变迁。设置完成后转入步骤 2。

2．建立服务信任连接

通过建立代表具体实体访问控制策略的下层 Petri 网，扩展上层 Petri 网中的各抽象变迁。客户实体欲访问服务集合中的服务，按照访问控制策略路径要求须提供信任证序列：

$$G=(C_1,\cdots,C_n)$$

Petri 网执行开始。由访问控制策略路径可知客户实体首先须提供的信任证：

$$G_1 = (C_i, C_j), 1 \leqslant i \leqslant j \leqslant n$$

则 Petri 网中消耗相应托肯，信任值达到阈值，变迁 1 发生并产生新托肯 T_1。执行继续，客户实体须在 T_1 的基础上提供信任证。

$$G_2 = (C_k, C_m), 1 \leqslant k \leqslant m \leqslant n$$

则 Petri 网中消耗相应托肯，信任值达到阈值，变迁 2 发生并产生新托肯 T_2。以此类推，直至客户实体提供全部所需信任证 $G = (C_1, \cdots, C_n)$，达到阈值以激发客户实体所需服务。此时信任关系建立成功并转步骤 3。

若客户实体不能提供某信任证，则服务实体尝试引导至其他可选路径；若所有路径均不可达，则信任关系建立失败，服务请求被拒绝。尝试连接其他实体，重复步骤 2。实体连接成功后，设置当前访问控制策略为该抽象变迁的扩展 Petri 网。若所有实体都无法正确连接，则系统初始化失败，须重新设置待连接实体。

3．服务进行中的可信性评估

服务访问初始化完成，服务进行首次成功执行。由专家知识得到系统可信性的先验信息 $P(D_S)$，算法启动。根据服务执行而生成的数据信息及审计日志，可计算得到证据信息 $P(M_i \mid D_S)$ 及 $P(M_i)$。问题转化为根据先验信息及证据信息计算后验信息。

根据贝叶斯公式可得此时系统可信性值：

$$P(D_s \mid M_i) = \frac{P(M_i \mid D_s) P(D_s)}{P(M_i)}$$

设当前服务激发安全性阈值为 P_E，则访问控制策略为

$$\begin{cases} \text{继续后续服务并进入步骤4,} & \text{当} P \geqslant P_E \text{时} \\ \text{转入步骤5,} & \text{当} P < P_E \text{时} \end{cases}$$

4．交互历史对可信性的反馈

不同的历史信息会对系统当前可信性值产生不同的反馈效果，即历史信息越近，反馈效果越大。设影响因子 ε 表示历史信息的反馈效果，则某一特定历史信息的反馈效果应随着协作活动的不断深入而衰减。当 ε 衰减到一定程度时，其所代表的历史信息将失去对实体可信性值的反馈效果。此时，应抛弃该历史信息。

使用滑动窗口来模拟该过程。为协作过程中的每一个客户实体建立一个滑动窗口来记录最近 n 次协作过程中历史信息的情况，如图 5.8 所示。

图 5.8　滑动窗口

图 5.8 中为一个 n 次带权滑动窗口，从左至右编号为 1,2,…,n–1,n，记录了最近 n 次实体的交互情况。滑动窗口是一个 FIFO 队列，窗口 i 被赋予了权值 ε_i，代表相应历史信息的反馈效果。当一次新的协作完成之后，交互历史信息从左侧进入滑动窗口，原有的记录依次右移，最右端记录被移出窗口队列并被丢弃。计算事实证据时，每次交互历史须加权后再进行计算，即

$$P(M) = \varepsilon_1 P(M_1) + \varepsilon_2 P(M_2) + \cdots + \varepsilon_n P(M_n)$$

从 ε_1 到 ε_n 的值依次递减，也即时间越久的历史信息影响因子越小。

滑动窗口模拟了历史信息的衰减过程，保证了最近发生的历史信息具有较大的反馈效果。滑动窗口个数 n 体现了评估者对于历史信息的重视程度，n 越大说明评估者对于历史信息越看重。

5．信任关系的演化

每次服务执行后，系统均计算当前可信性值，并实时监控其变化。对于所有服务 S_i，设其激发可信性阈值为 P_{S_i}，访问控制策略为

$$\begin{cases} \text{继续后续服务}, & \text{当} P \geqslant P_{S_i} \text{时} \\ \text{拒绝后续服务}, & \text{当} P < P_{S_i} \text{时} \end{cases}$$

当前服务被拒绝时，客户实体可有两种选择：继续请求此服务，则转入步骤 1，重新进行信任连接的建立；选取其他服务实体，进入如下结构演化策略。

设当前需要进行结构演化的实体为目标实体。

（1）所有需要目标实体参与事务的其他实体均进入被动状态。

（2）目标实体向服务实体请求建立连接。

（3）连接建立成功，修改上层 Petri 网结构及下层 Petri 网中的访问控制策略。

（4）所有相关实体恢复主动状态。

下面介绍系统结构建模实例。按照前述原则，对于图 5.1（c）所示的相依性树进行建模，并根据变迁 t_4 的访问控制策略构造其下层 Petri 网。上层 Petri 网建模如图 5.9 所示。

实体 A_2 的运行需要实体 B_2，B_3 提供服务，是一对多的关系。变迁 t_1 设置为要求较高的实体 B_2 的访问控制策略，若此策略不满足，则系统必不能获得执行。实体 B_2 的执行需要实体 C_1 或 C_3 提供服务，

是一对多或的关系。系统执行时，变迁 t_1 具有执行权，若执行成功，则消耗库所 A_2 中的托肯，在库所 B_2、B_3 中各生成一个托肯。此时对于实体 B_2，变迁 t_2 和 t_3 都有可能发生，根据顺序首先执行 B_2 与 C_1 的协商，若成功则执行 t_2，否则尝试执行 t_3。同时 t_4 变迁执行，成功后消耗库所 B_2、B_3 中的托肯，在 C_1，C_2，C_3 中各生成一个托肯。当所有输出库所中都包含托肯时，说明系统可执行成功。

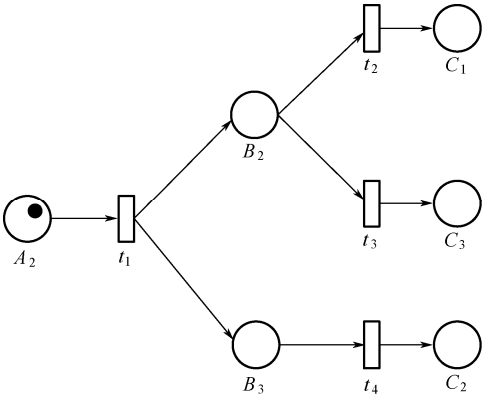


图 5.9　Petri 网相依性树建模

通过对系统的可达性、有界性、活性等各种动态特性进行分析来验证 Petri 网模型，这种分析可通过可达树的构造和正确性检查实现。可达树的基本思想就是将可到达的标识作为节点，变迁的触发作为连接弧，来构造一棵树。在构造树的过程中，检查服务组合的状态和其中库所的托肯数目以实现流程模型的验证。

对于文中的分层 Petri 网，可构造上层可达树，如图 5.10 所示，$M_i=(A_2,B_2,B_3,C_1,C_3,C_2)$。

由可达树可以看出，对于系统运行中可能出现的全部状态的集合 $R(M_0)$，以及库所集合 S，$s\in S$，存在正整数 1，使得 $\forall M\in R(M_0):M(s)\leqslant 1$，

说明该系统有界。对于变迁集合 T，$t \in T$，都存在 $\forall M' \in R(M_0)$，使得 $M'[t>$，说明该 Petri 网是活的。Petri 网中每次触发都将推向终态，不会出现无限的循环，则该 Petri 网具备前进性，且该 Petri 网所有状态都是可达的，该 Petri 网具备完整性。

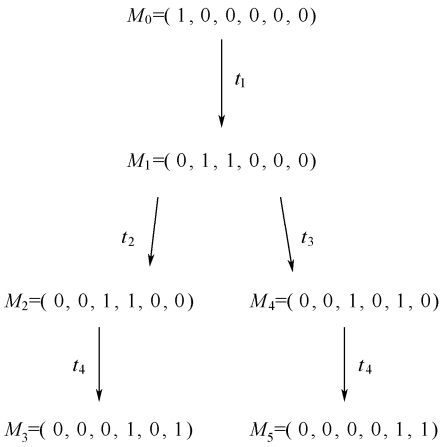


图 5.10　上层可达树

下面介绍契约协商策略建模实例。设服务实体的访问控制策略为

$$R \leftarrow (C_1 \wedge C_6) \vee (C_2 \wedge C_3), R_1 \leftarrow (C_2 \wedge C_3) \vee (C_2 \wedge C_4), R_2 \leftarrow C_7, R_3 \leftarrow C_5 \vee C_4$$

则下层 Petri 网模型如图 5.11 所示。

客户实体欲访问服务集合 R 中的服务，可提供信任证 C_2、C_3。若无法提供，则按照 Petri 网路径要求须先达到库所 R_4、R_5；若要达到此条件，则须先达到库所 R_1、R_2，或 R_1、R_3，或 R_2、R_3；以此类推，由系统引导客户提供相应信任证序列，直到有一条可达路径或全部不可达。此下层 Petri 网可达树构造为 M_i=(C_3,C_2,C_4,C_5,C_7,R_1,C_1,R_3,C_6,R_2,R_4,R_5,R)，如图 5.12 所示。

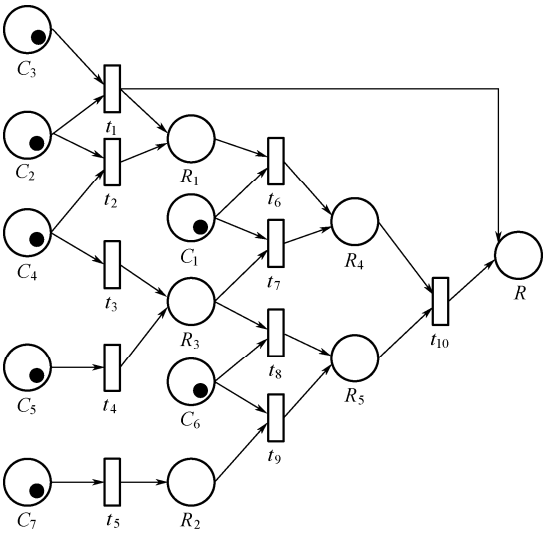


图 5.11　基于 Petri 网的信任契约协商模型

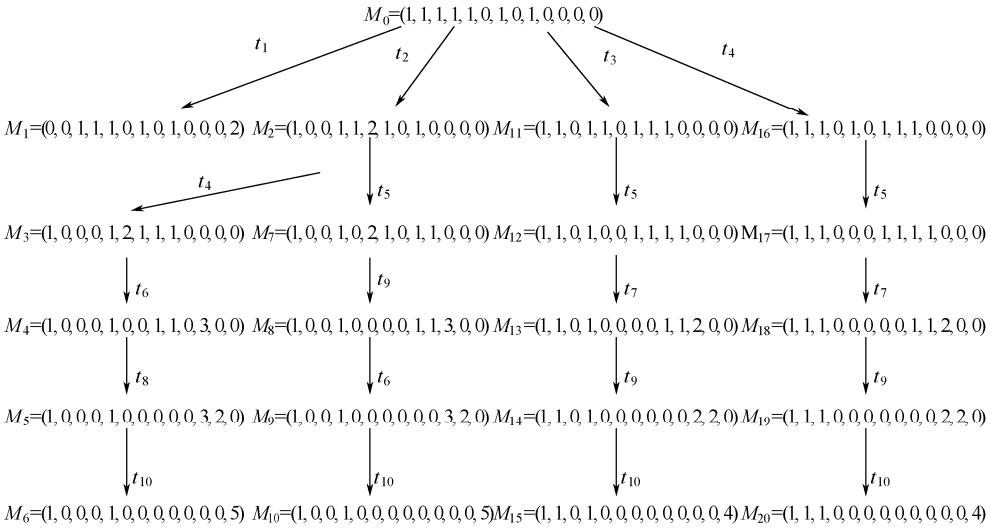


图 5.12　下层可达树

使用可达树中的数值表示库所中托肯权值之和，为简便起见，所有托肯权值均设为 1;所有变迁激发阈值都设为输入库所中托肯权值之和，

即只要达到变迁所需信任证即可激发变迁。由可达树可以看出，对于系统运行中可能出现的全部状态的集合 $R(M_0)$，以及库所集合 S，$s \in S$，存在正整数 5，使得 $\forall M \in R(M_0): M(s) \leqslant 5$，说明该系统有界。对于变迁集合 T，$t \in T$，都存在 $\forall M' \in R(M_0)$，使得 $M'[t>$，说明该 Petri 网是活的。Petri 网中每次触发都将推向终态，不会出现无限的循环，则该 Petri 网具备前进性，且该 Petri 网所有状态都是可达的，该 Petri 网具备完整性。

第 6 章

网构软件可信性评估模型

可信性作为评估软件系统性能的指标，通常包含可用性、可靠性、防危性、安全性、可维护性等特征属性[167]。目前针对这些特性属性的研究已取得了较多成果，可以较为准确地评价软件系统某一方面或几方面的特性。但对于网构软件来说，不同用户的具体情况不同，对软件系统可信性各方面特性指标的要求也不尽相同，这就需要一个统一的可信性指标对系统的各方面特性指标进行综合评价。而通过各方面特性综合评估软件系统可信性的研究还不多，且多是使用传统的软件可靠性评估模型。本章基于第 1 章所述软件系统可信性特征属性的概念，并综合 ISO/IEC 9126[98]及 SJ/T 11374—2007[168]标准中对软件质量属性的划分，对基于贝叶斯网络的网构软件可信性评估方法进行研究。在第 5 章网构软件体系结构分析的基础上，建立系统结构模型，根据此模型建立基于贝叶斯网络的网构软件的可信性评估体系。该评估体系从可信性与其特征属性之间的关系，以及网构软件的整体系统与其各组成实体之间的关系入手，使用贝叶斯网络对网构软件可信性的各特征属性进行综合评估，同时为用户对第三方实体做出最优选择方案提供良好的依据。

6.1　基于贝叶斯网络的网构软件可信性评估体系

软件可信性评估指标体系是一套能够全面反映网构软件可信特征，并且具有内在联系，起互补作用的指标集合。软件可信性评估不仅包含软件可靠性、安全性、效率等对传统软件产品的度量，还应包含对软件开发方法、软件管理、软件配置等与产品相关因素的度量，从而保证评价结果的可信性。同时，网构软件可信性评估体系还应有针对网构软件特征的度量指标。本节在第 5 章网构软件结构分析的基础上提出的基于贝叶斯网络的网构软件可信性评估指标体系模型，融合了传统指标与网构软件特定指标，涵盖了动态指标与静态指标，为网构软件的设计、开发、部署、评估奠定了良好的基础。

6.1.1　贝叶斯网络

贝叶斯网络（Bayesian Network）[169]又称贝叶斯信念网（Belief Network），它采用图形化的网络结构直观地表达变量的联合概率分布及其条件独立性。

定义一个贝叶斯网络[170]为一个有向无环图（Directed Acyclic Graph，DAG），由代表变量的节点及连接这些节点的有向边构成，有向边由父节点（双亲节点）指向子节点（后代节点），用单线箭头“→”

表示。

设 $V=\{X_1,X_2,\cdots,X_n\}$ 是值域 U 上 n 个随机变量，则值域 U 上的贝叶斯网络为 $BN(B_s,B_p)$，其中：

（1）$B_s=(V,E)$ 是一个定义在 V 上的有向无环图 Γ，V 是该有向无环图 Γ 的节点集，E 是 Γ 的边集；如果存在一条节点 X_i 到节点 X_j 的有向边，则称 X_i 是 X_j 的父节点，X_j 是 X_i 的子节点；记 X_j 的所有父节点为 πX_j。

（2）$B_p=\{P(X_i\,|\,\pi X_i)[0,1]\,|\,X_i\in V\}$，对于 V 中的每个节点，定义了一组条件概率分布函数 $P(X_i\,|\,\pi X_i)[0,1]$。

给定一个有向无环图 Γ 和一个离散变量集合 $V=\{X_1,X_2,\cdots,X_n\}$ 上的联合概率分布 P，如果 Γ 可以代表 P，即在 X 中的变量和 Γ 的节点之间存在一一对应的关系，使得 P 可以进行如下的递归乘积分解：

$$P(X)=\prod_{i=1}^{n}P(X_i\,|\,\pi X_i)$$

其中，πX_i 是图 Γ 中 X_i 的直接父节点，则将图 Γ 和概率分布 P 的联合称为贝叶斯网络[171]。

关于一组变量 $V=\{X_1,X_2,\cdots,X_n\}$ 的贝叶斯网络由以下两个部分组成[172]：

① 一个表示 V 中变量的条件独立断言的网络结构；

② 与每一个变量相联系的局部概率分布集合 P。

贝叶斯网络具有如下特性[170,171,173]：

（1）条件独立性，即贝叶斯网络规定图中的每个节点 V_i 条件独立于由 V_i 的父节点给定的非 V_i 后代节点构成的任何节点子集；也就是说，如果用 $A(V_i)$ 表示非 V_i 后代节点构成的任何节点子集，用 $pa(V_i)$ 表示 V_i 的直

接双亲节点，则 $I(V_i, A(V_i) \mid pa(V_i))$。

其意义为

$$P(V_i \mid A(V_i), pa(V_i)) = P(V_i \mid pa(V_i))$$

由于此性质，在求变量的概率信息时，只需要考虑与该变量有关的有限变量，可以大大降低问题的求解难度，从而使许多复杂问题得到可行的解决方法。

（2）基于概率论的严格推理。贝叶斯网络是一种不确定性知识表达与推理模型，它的推理原理基于 Bayes 概率理论，推理过程实质上就是概率计算。

（3）知识表示分为定性知识和定量知识，定性知识指网络的结构关系，表达事件之间的因果联系；定量知识指节点的条件概率表，主要来源于专家经验、专业文献和统计学习。

（4）知识获取与推理的复杂度较小。由于贝叶斯网络具有条件独立的特点，因此可以降低知识获取与推理的复杂程度。在知识获取时，只要关心与节点相邻的局部网络图；在推理计算时，只要已知节点的相关节点的状态即可估计该节点的概率信息。

在进行贝叶斯网络的计算时，还涉及以下定义[174]：

- 先验概率。先验概率是指根据历史资料或主观判断所确定的各种事件发生的概率，该概率没能经过实验证实，属于检验前的概率，称之为先验概率。

- 后验概率。后验概率一般是指利用贝叶斯公式，结合调查等方式获取了新的附加信息，对先验概率进行修正后得到的更符合实际的概率。

- 联合概率。联合概率也叫乘法公式，是指两个任意事件的乘积的

概率，或称之为交事件的概率。

- 全概率公式。如果影响事件 A 的所有因素 $B_1, B_2, \cdots, B_n$ 满足 $B_i \bullet B_j = \Phi, i \neq j$ 且 $P(\cup B_i) = 1, P(B_i) > 0, i = 1, 2, \cdots$，则必有 $P(A) = \sum P(B_i) P(A \mid B_i)$。

- 贝叶斯概率。贝叶斯概率是通过先验知识和统计现有数据，使用概率的方法对某一事件未来可能发生的概率进行估计。

贝叶斯网络理论的研究在实际应用中发挥了巨大作用并体现了深厚潜力。目前贝叶斯网络理论已成功应用到故障诊断[175]、军事决策[176]、智能机器人[177]、医学上的病理诊断[178]、信息融合[179]、信息智能检索[180]等领域。

在软件可信性评估方面，目前研究较多的领域是使用贝叶斯网络进行复杂系统可靠性建模。这方面最初的研究是 Torres-Toledano，Sucar[181]及 Arroyo 等[182]通过使用贝叶斯网络估计错误相关性概率来为复杂系统的可靠性进行建模。之后的研究覆盖了可信性的若干方面。Bai 等[183]提出使用马尔科夫链的扩展贝叶斯网络通过评估离散时间错误数据进行考虑系统使用情况下的可靠性预测。Axel 与 Helminen[184]使用贝叶斯网络合并的方法，在软件安全背景下进行基于数字系统软件的可靠性估计。Boudali 与 Dugan[185]提出基于贝叶斯网络形式化方法的动态系统可靠性建模与分析体系。Bouissou[186]等对专家知识进行形式化以便评估关键系统可信性的多方面属性。Montani 等[187]设计出将动态错误树（DFT）转换为动态贝叶斯网络（DBN）的工具，将动态方面的特性整合进了可靠性评估中。Weber 等[188]提出考虑受外部因素约束的系统退化或错误模式的系统可信性建模方法，外部因素主要由维护操作、生产情况、环境情况等系统使用情况产生。Boudali 和 Dugan [189] 针对动态系统可靠性建模与分析提出连续时间贝叶斯网

络（CTBN）体系，使用带有采样时间的连续节点来对构件的错误分布进行建模。

目前，在可信性的一个属性即可靠性评估方面，已有了较多深入的研究成果，且多将系统考虑为一个整体。本书提出的可信性评估模型将可信性的多个基础属性进行综合，由网构软件的各基础实体自底向上计算，最终得到系统整体可信性。并且将系统的技术、组织、决策、人力等主观方面的因素考虑在内形成一个统一的模型，这是前述研究中都没有考虑到的。

6.1.2　网构软件可信性评估体系

作为一种软件系统，网构软件的可信性受到诸多因素的影响，应由多个特征属性指标综合而成。同时，网构软件是一种分布在广域范围内、由第三方提供的一组软件实体的联盟，其中每一个基本实体首先都是一个构件，以确保单个实体可以独立部署、运行与演化，因此适用于构件的可信性评估模型。但网构软件的自主性、演化性、协同性、多态性和反应性等特性决定了其基本实体又不能完全等同于构件。互联网“天然”的异构性决定了网构软件基本实体所具备的特性不尽相同，其可信性评估还应包括特有的指标。这里基于可信性特征属性的概念建立可信性评估体系。参照 ISO/IEC 9126 标准对软件质量特性的分析，将软件可信性特性进行分解，并加入对软件构件区别于一般软件的特殊质量特性——可复用性的度量。同时根据网构软件的特性，建立树形结构的可信性评估指标体系，如图 6.1 所示。最下层指标没有全部列出，仅举几例进行说明。

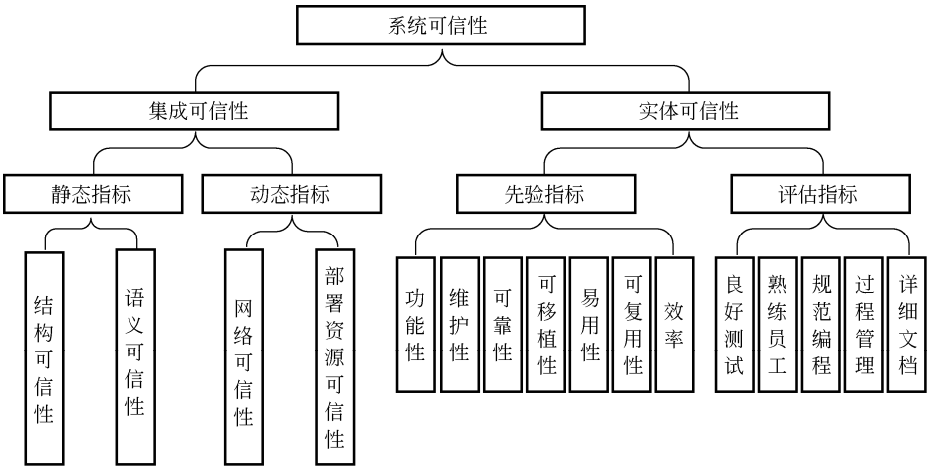


图 6.1　网构软件可信性评估体系

定义 6.1（系统整体可信性）网构软件系统整体的可信性，用以评估整个系统的可信性。位于网构软件可信性评估体系的最顶层，可将系统可信性划分为实体可信性与集成可信性。

定义 6.2（实体可信性）针对单个实体的评估指标，在网构软件中可将各实体看做单独的构件进行评估。其中包括先验指标和评估指标。

定义 6.3（先验指标）对实体可信性的初始评估，由功能性、可靠性、易用性、效率、维护性、可移植性和可复用性等传统可信性指标构成，可采用传统方法计算获得。

定义 6.4（评估指标）通过对与软件的可信性息息相关的软件开发方法、软件管理、软件配置因素进行评估，对前面获得的先验指标进行修正，以使可信性的度量更加精确、客观。此指标的计算数据可通过统计获得，作为评估系统的训练数据集。

定义 6.5（集成可信性）针对集成后系统而言的可信性指标，着重考虑两个或多个实体组装在一起后对整体可信性的影响。由于实体组装

后的可信性并不是几个单独实体可信性的简单叠加，因此需要通过若干指标进行修正。这些指标涵盖静态指标和动态指标两方面。

- 静态指标。重点从系统体系结构方面评估实体间的结构相依性与语义相依性，也可称之为设计时指标。由于实体间接口一致性、协议兼容性等对组装后实体集合的服务质量有极大影响，因此，这些指标应作为评估集成可信性的重要因素，并以此修正单个实体的可信性结果。

- 动态指标。网构软件开发的另一重要环节是部署与运行，此时应重点考虑软件部署与运行时的安全性、稳定性等动态指标，也可称之为运行时指标。这些指标对实体集合的服务质量同样有较大的影响。例如，如果到某实体的网络稳定性较差，则该实体很难完成正常的服务。因此，动态指标应从另一方面来修正单个实体的可信性结果。

综上所述，系统评估从静态、动态两方面进行，既包含了整体与个体的对比，又涵盖了静态与动态的特征。同时，指标集还具有良好的可扩展性，当需要新增指标或对原有指标进行改动时，只需要在指标体系中对相应的指标进行添加、删除、替换，而整体的计算方法不变。由此可见，本书所研究的可信性评估体系不但可用于度量网构软件的可信性，对其设计与部署也有着明确的指导意义。网构软件在运行过程中也可动态评估各实体的可信性情况，随时切换到可信性更高的实体以保证服务质量，充分体现了网构软件的自主性、演化性、协同性、多态性和反应性等特性。

6.1.3　基于贝叶斯网络的网构软件可信性评估

对于网构软件系统来说，系统的可信性是由各实体自身可信性及其

连接子的可信性组成的。基于第 5 章中所述的网构软件结构模式，从相依性矩阵中选取元素 $a_{i,j}=1$ 的实体，对图 5.1 中的各相依性树建立贝叶斯网络，并将指标体系用在此结构中，对网构软件系统进行评估。评估场景如图 6.2 所示。

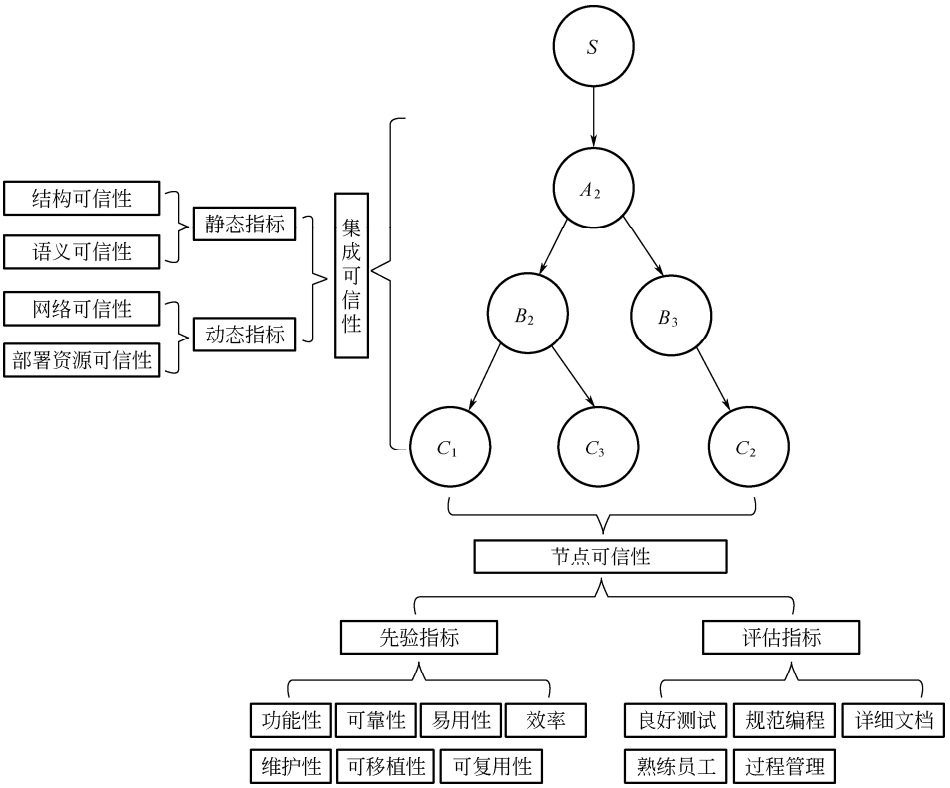


图 6.2　网构软件可信性评估体系评估场景

贝叶斯网络表示一组变量的联合概率分布。图 6.2 中显示了实体 A_2，B_2，B_3，C_1，C_2，C_3 可信性的联合分布。一般地说，贝叶斯网络表示联合概率分布的方法是指定一组条件独立性假设，以及一组局部条件概率集合。联合空间中每个变量在贝叶斯网中表示为一个节点，需要两种类型的信息。首先，网络弧表示断言“此变量在给定其直接前驱时条件独

立于其非后继”。如图 6.2 所示，从 B_2 到 C_3 存在一条有向路径，称 C_3 为 B_2 的后继。其次，对每个变量有一个条件概率表，它描述了该变量在给定其立即前驱时的概率分布。

可以用贝叶斯网络在给定其他变量如 C_3 的观察值时推理出目标变量如 B_2 的值，也即目标变量的概率分布，它指定了在给予其他变量观察值的条件下，目标变量取每一个可能值的概率，在这里即实体的可信性。在网络中所有其他变量（即该实体的指标值及所依赖的实体的可信性）都确切知道之后，这一推理步骤是很简单的。

因此，网构软件系统可信性计算方法如下：

（1）基础实体可信性计算。首先对单个基础实体进行评估，通过多种途径或方法获取先验指标，并通过训练数据得出的评估指标加以修正，可得到节点的可信性。

（2）上层实体可信性计算。依赖基础实体的上层实体同样需要对实体可信性进行计算，同时还需要使用各实体间的集成可信性指标进行修正，以保证上层实体可信性的精确性。集成可信性分为动态指标和静态指标，可分别计算得出系统动态方面的可信性和静态方面的可信性。

（3）迭代过程。对各上层实体重复步骤（2）进行迭代计算，上层实体可信性依赖下层实体，直到最后得出顶层实体的可信性。我们在顶层实体的层次之上再增加一层节点，即整个系统的可信性，可由综合顶层实体可信性，并通过实体间集成可信性指标修正计算得到。

（4）取最优解。当系统结构模式可划分为多棵相依性树，即系统的组成可有多种选择时，我们可对每棵相依性树都建立贝叶斯网络，并计算最终的系统整体可信性，以便做出最佳选择。

6.2 可信性评估指标计算方法

我们基于网构软件的结构模式定义了其静态可信性指标及动态可信性指标，将可信性看做系统满足可信性指标时能提供可信服务的条件概率。设可信性指标 $P(M)=\{P(M_i)\mid i=1,2,\cdots,n\}$，则系统可信性 $P(D_s\mid M_i)$ 可通过贝叶斯推理计算：

$$P(D_s\mid M)=\frac{P(M\mid D_s)P(D_s)}{P(M)} \tag{6.1}$$

贝叶斯推理基于如下假定：待考察的量遵循某概率分布，且可根据这些概率及已观察到的数据进行推理，以做出最优的决策。使用贝叶斯方法时，观察到的每个训练样例可以增量地降低或升高某假设的估计概率，提供了一种比其他算法更合理的学习途径；先验知识可以与观察数据一起决定假设的最终概率，其形式可以是每个候选假设的先验概率，或者每个可能假设在可观察数据上的概率分布。上层元素可以由多个假设一起做出预测，用它们的概率来加权。

因此，要计算系统整体可信性，首先需要计算各可信性指标的值。

6.2.1 静态可信性指标

静态可信性指标主要从系统结构相依性及语义相依性方面进行度量，面向系统设计阶段，根据不同实体的选择，对组装后的系统的可信性进行评估。

1．实体可信性

网构软件中的各实体可看做单独的构件，其可信性可利用构件可信性指标（如功能性、可靠性、易用性、效率、维护性、可移植性和可复用性等）进行评估，同时使用其质量对于实体满足质量标准要求的条件概率进行修正，即实体在开发、测试、部署等过程中按照质量标准执行后其服务可信的概率对自身可信性的影响，如图 6.3 所示。

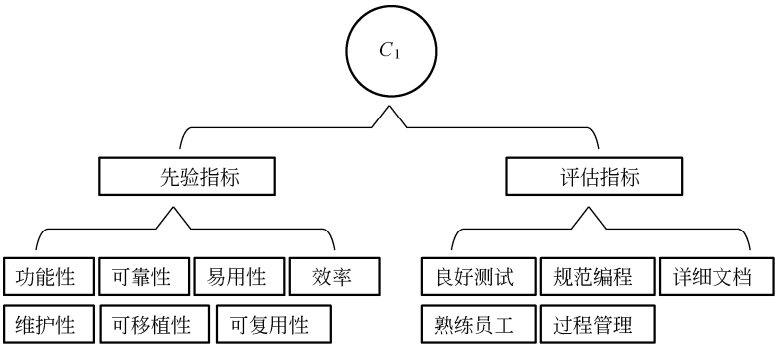


图 6.3　实体可信性指标

实体 C_1 的可信性可由其先验指标计算，同时使用评估指标进行修正。设 C_1 的可信性为 $P(D_{C_1} \mid M)$，由式（6.1）得

$$P(D_{C_1} \mid M) = \frac{P(M \mid D_{C_1})P(D_{C_1})}{P(M)} \tag{6.2}$$

其中，$P(D_{C_1})$ 是先验指标，即由功能性、可靠性、易用性、效率、维护性、可移植性和可复用性等传统可信性指标构成的对实体可信性的初始评估。$P(M \mid D_{C_1})$ 与 $P(M)$ 都可通过评估指标获得。用来计算可信性值的统计数据必定只是所有指标数据的子集，因此，设各先验指标为 $P(M_p) = \{P(M_{pi}) \mid i = 1,2,\cdots,n\}$，各评估指标为 $P(M_e) = \{P(M_{ei}) \mid i = 1,2,\cdots,n\}$。

式（6.2）中的 $P(D_{C_1}|M)$ 可由下式近似计算：

$$P(D_{C_1}|M)\approx\frac{P(M_e|D_{C_1})P(M_p)}{P(M_e)} \tag{6.3}$$

2．集成可信性

在网构软件系统中，上层实体的可信性还应考虑其基础实体可信性的影响，即其所依赖的基础实体由实体间的结构相依性和语义相依性所产生的影响，通过集成可信性指标表示，如图 6.4 所示。

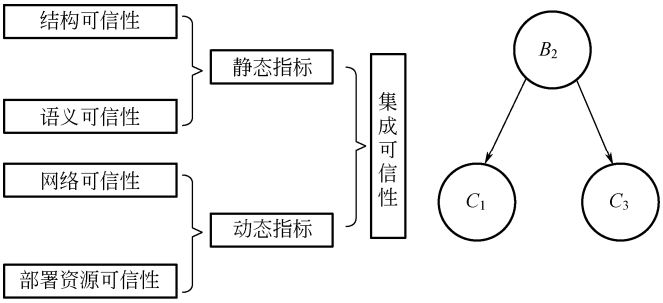


图 6.4　集成可信性指标

实体 B_2 的可信性除了通过自身的实体可信性计算外，还应通过实体 C_1 和 C_3 的条件概率进行修正。

设 B_2 可信性为 $P(D_{B_2}|M)$，依赖实体可信性 $P(D_{Dp})=\{P(D_{Dpi})|i=1,2,\cdots,n\}$，实体间结构可信性为 $P(D_{St})=\{P(D_{Sti})|i=1,2,\cdots,n\}$，实体间语义可信性为 $P(D_{Se})=\{P(D_{Sei})|i=1,2,\cdots,n\}$，计算方法如下：

$$P(D_{B_2}|M_p,M_e,D_{St},D_{Se},D_{Dp})=\frac{P(M_e,D_{St},D_{Se},D_{Dp}|D_{B_2})P(M_p)}{P(M_e,D_{St},D_{Se},D_{Dp})} \tag{6.4}$$

其中，结构可信性为各实体间结构相依性的可信性，即系统结构方面的可信性，如语言兼容性、接口一致性等。语义可信性为各实体间语义相依性的可信性，即系统语义方面的可信性，如数据复制一致

性、故障检测、故障恢复等。该贝叶斯网络是由相依性矩阵中元素 $a_{i,j}$=1 的实体构成的，因此结构可信性和语义可信性值均为 1，则式（6.4）可简化为

$$P(D_{B_2} \mid M) = \frac{P(M_e \mid D_{B_2})P(M_p)}{P(M_e)} P(D_{B_2} \mid D_{Dp}) \tag{6.5}$$

$P(D_{B_2} \mid D_{Dp})$ 也即基础节点可信时 B_2 的可信性，系统投入运行前 $P(D_{B_2} \mid D_{Dp})$ 的值可设为某一初值，如 $P(D_{B_2})$。当系统运行一段时间后，可根据具体运行情况统计出证据数据进行修正。

6.2.2　动态可信性指标

由于网构软件所处环境的开放性、动态性、难控性，在部署及运行过程中随时可能出现各种各样导致可信性下降的情况，因此，网构软件可信性评估中还应包括如网络环境可信性、部署资源可信性等动态可信性指标。通过动态可信性指标对实体当前的服务质量进行评估，当发现可信性下降时，系统可从当前实体切换到功能相同且可信性较高的实体，充分体现了网构软件的自主性、演化性、协同性、多态性和反应性等特性。

同样以图 6.4 中的实体 B_2 为例。实体 C_1、C_3 可信性对 B_2 的影响需要通过其动态指标进行加权修正。动态指标包括网络环境可信性、部署资源可信性等。设实体 B_2 的动态可信性为 D'_{B_2}，网络环境可信性为 D_N，部署资源可信性为 D_R，计算方法如下：

$$P(D'_{B_2} \mid M_p, M_e, D_N, D_R) = \frac{P(M_e, D_N, D_R \mid D'_{B_2})P(M_p)}{P(M_e, D_N, D_R)} \tag{6.6}$$

其中，D_N为实体 B_2所依赖实体的实体可信性与其网络环境可信性加权后的并集，D_R为实体 B_2所依赖实体的实体可信性与其部署资源可信性加权后的并集，即

$$D_N = \bigcup_{i=1}^{n} \left(D_{Ni} \bullet D_{pi} \right)$$

$$D_R = \bigcup_{i=1}^{n} \left(D_{Ri} \bullet D_{pi} \right)$$

网络环境可信性根据实体间网络情况对其可信性进行度量评估，影响因素包括带宽、延迟等。部署资源可信性对于实体所要部署的系统环境的可信情况进行评估，以避免系统的漏洞对整体可信性造成影响。此类可信性的评估方法将在下章进行详细介绍。

错误注入技术[190,191]是按照选定的错误模型，用人工方法有意识地产生错误并施加于运行特定工作负载的目标系统中，以加速该系统错误和失效的发生，同时观测和回收系统对所注错误的反应信息，并对回收信息进行分析，从而向试验者提供有关结果的试验过程。该技术近几年在各种测试领域得到了广泛的研究[192~197]。

6.2.3 系统整体可信性计算方法

在最上层实体的层次之上再增加一层节点，即整个系统的可信性。从最底层实体的可信性开始计算，逐层上推，最终可计算出最上层实体的可信性，而整个系统的可信性由最上层各实体可信性决定，如图 6.5 所示。

按照前述方法可计算出基础实体 C_1，C_2，C_3的可信性，实体 B_2的

可信性可通过 C_1、C_3 计算得出，实体 B_3 的可信性可由实体 C_2 计算得出。同理可由 B_2、B_3 的可信性计算出实体 A_2 的可信性，最后得出系统整体可信性。计算方法如下。

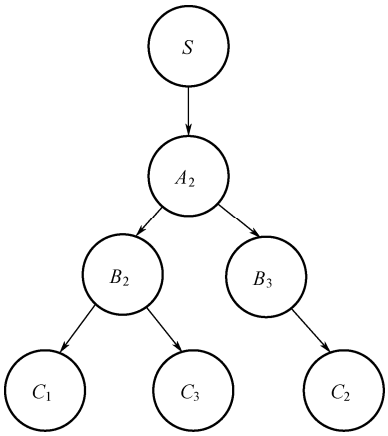


图 6.5　系统整体可信性计算

（1）各基础实体可信性

$$C_1:\quad P(D_{C_1}\mid M)=\frac{P(M_e\mid D_{C_1})P(M_p)}{P(M_e)}$$

$$C_2:\quad P(D_{C_2}\mid M)=\frac{P(M_e\mid D_{C_2})P(M_p)}{P(M_e)}$$

$$C_3:\quad P(D_{C_3}\mid M)=\frac{P(M_e\mid D_{C_3})P(M_p)}{P(M_e)}$$

（2）第二层实体可信性

- 静态指标

$$B_2:\quad P(D_{B_2}\mid M)=\frac{P(M_e\mid D_{B_2})P(M_p)}{P(M_e)}P(D_{B_2}\mid D_{C_1})P(D_{B_2}\mid D_{C_3})$$

$$B_3:\quad P(D_{B_3}\mid M)=\frac{P(M_e\mid D_{B_3})P(M_p)}{P(M_e)}P(D_{B_3}\mid D_{C_2})$$

- 动态指标

$$B_2:\quad P(D'_{B_2}\mid M_p,M_e,D_N,D_R)=\frac{P(M_e\mid D'_{B_2})P(M_p)}{P(M_e)}P(D'_{B_2}\mid D_N)P(D'_{B_2}\mid D_R)$$

$$B_3:\quad P(D'_{B_3}\mid M_p,M_e,D_N,D_R)=\frac{P(M_e\mid D'_{B_3})P(M_p)}{P(M_e)}P(D'_{B_3}\mid D_N)P(D'_{B_3}\mid D_R)$$

其中，D_N、D_R即为其所依赖的C_1、C_3节点到B_2节点的网络环境可信性和部署资源可信性与C_1、C_3实体可信性加权后的并集，即

$$D_N=(D_{N1}\bullet D_{C_1})\cup(D_{N3}\bullet D_{C_3})$$

$$D_R=(D_{R1}\bullet D_{C_1})\cup(D_{R3}\bullet D_{C_3})$$

对于B_3节点同理计算。

（3）顶层实体可信性

- 静态指标

$$A_2:\quad P(D_{A_2}\mid M)=\frac{P(M_e\mid D_{A_2})P(M_p)}{P(M_e)}P(D_{A_2}\mid D_{B_2})P(D_{A_2}\mid D_{B_3})$$

- 动态指标

$$A_2:\quad P(D'_{A_2}\mid M_p,M_e,D_N,D_R)=\frac{P(M_e\mid D'_{A_2})P(M_p)}{P(M_e)}P(D'_{A_2}\mid D_N)P(D'_{A_2}\mid D_R)$$

其中，D_N、D_R即为其所依赖的B_2、B_3节点到A_2节点的网络环境可信性和部署资源可信性与B_2、B_3实体可信性加权后的并集，即

$$D_N=(D_{N2}\bullet D_{B_2})\cup(D_{N3}\bullet D_{B_3})$$

$$D_R=(D_{R2}\bullet D_{B_2})\cup(D_{R3}\bullet D_{B_3})$$

（4）系统整体可信性

$$P(D_s\mid M)=\frac{P(M_e\mid D_s)P(M_p)}{P(M_e,D_{A_2})}P(D_s\mid D_{A_2})$$

系统整体可信性的动态指标与静态指标都可依此计算。

当系统有多种组装方案可选时，可全部计算各种情况后综合选取最优方案。

6.3　网构软件可信性评估实例及结果分析

以第 4 章软件评测中心实验场景为基础，模拟了一个实际的软件测试系统对上述研究成果进行实验。系统结构如图 6.6 所示。

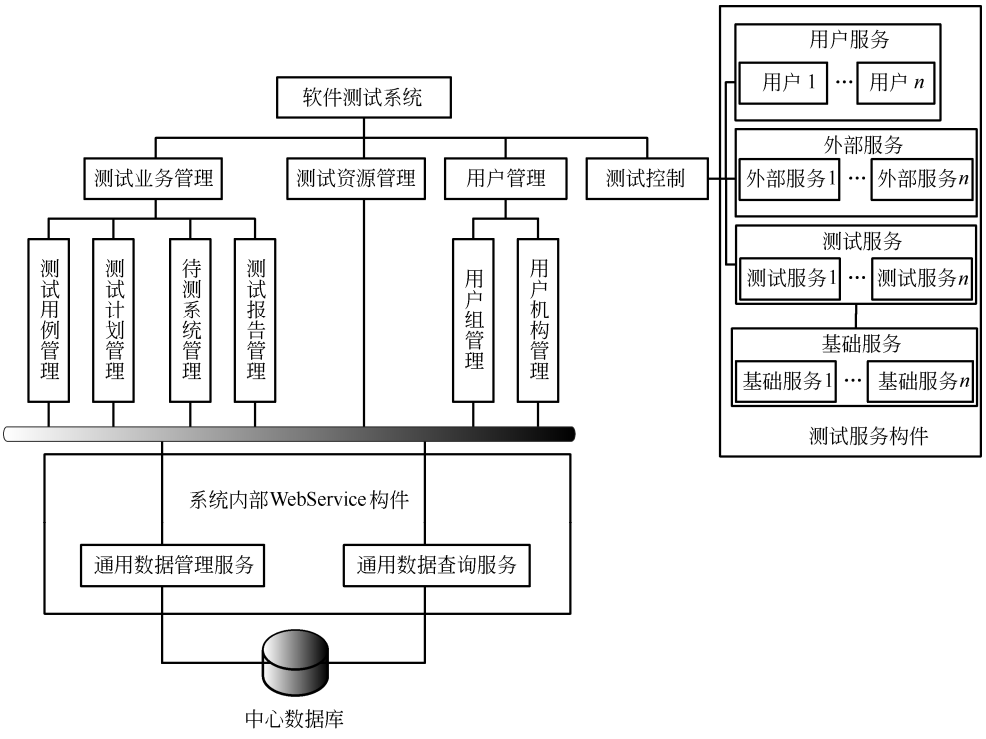


图 6.6　软件测试系统结构

系统总体可分为三部分，即业务逻辑模块、系统内部构件和测试服务构件。业务逻辑模块即评测中心为完成特定的管理功能而开发的系统模块，使用常见的编程语言开发；系统内部构件是评测中心为提高系统的复用度、降低开发成本，将部分通用的功能封装成可直接调用的构件，通常为本地构件或网络服务形式；测试服务构件用于执行实际的测试服务，包括对本地测试软件封装后的测试服务、由第三方提供的外部服务、与测试用户系统进行交互的用户方服务。系统内部构件采用网络服务形式，通过标准 WebService 接口同业务逻辑模块进行交互。外部服务及用户服务以服务或分布式对象的形式通过互联网与本地系统进行交互，一般为较为成熟且封装良好的实体单元，是网构软件系统的主要组成元素。

6.3.1 待评估系统结构分析

采用自底向上的方式对软件系统各模块进行结构分析，以便得到系统的结构相依性图。测试服务模块是对外部服务的组合，需要与第三方实体进行交互，故首先进行分析。该模块主要用于执行实际的测试功能，共组合了 4 类服务接口。

（1）用户服务：组合各测试用户所提供的其企业内部信息系统接口。通过接口调用获取测试用户信息及待测系统的列表、描述信息等，供测试服务随时调用。

（2）外部服务：调用第三方测试服务提供商提供的服务，例如评测中心与测试工具提供商进行合作，由该提供商建立外部服务以代替在本地安装测试软件。

（3）测试服务：执行具体的测试服务，如单元测试、压力测试、安全性测试等，由测试控制构件根据当前待测系统的测试需求调用执行。

（4）基础服务：评测中心在进行测试时可能会需要与其他评测中心或服务提供商进行协同，以便获取基础数据或具体业务的支持。例如，在功能测试或性能测试中需要大量的测试数据，而专业的数据生成工具或大量的真实数据往往只被少数机构拥有，因此需要在测试过程中进行服务间的协同以获取这些数据。

在服务交互与远程对象调用方面，目前存在着多种技术协议，如 WebService、CORBA、Java-RMI 等。而这些协议之间又不能很好地兼容，因此采用不同协议的服务实体间就产生了结构相依性及语义相依性的差异。

外部服务构件中还包括了外部服务和用户服务，其中也涉及由于接口协议不兼容而引起的结构相依性及语义相依性问题，可按上述方法进行分析。

对于系统中的本地模块，如测试业务管理、测试资源管理、用户管理等，由于是采用同一开发环境开发并集成在一起使用的，相互关系较为紧密，且不存在接口兼容性问题，因此在结构分析时可作为同一实体对待。系统内部的通用构件可作为单独的服务实体，由于是自行开发的，与其他模块完全兼容，因此也不涉及结构相依性问题，只是作为系统可信性计算的一部分存在。最终可得如图 6.7 所示的整个系统结构相依性图。

在图 6.7 中，Rational Robot、Load Runner、JMeter 都是使用测试工具封装成的测试服务。其中 Rational Robot、Load Runner 只支持 WebService，而 JMeter 只支持 Java-RMI 协议。数据生成服务 1、数据

生成服务 2 只支持 WebService，数据生成服务 3 只支持 Java-RMI 协议。这样，当选用 Rational Robot 和 Load Runner 作为测试服务时，可选用数据生成服务 1 和数据生成服务 2 作为基础服务；而选用 JMeter 时，基础服务只能选用数据生成服务 3。后面的工作就是根据各服务的多方面属性计算组合后的可信性，以便采用最好的组合。

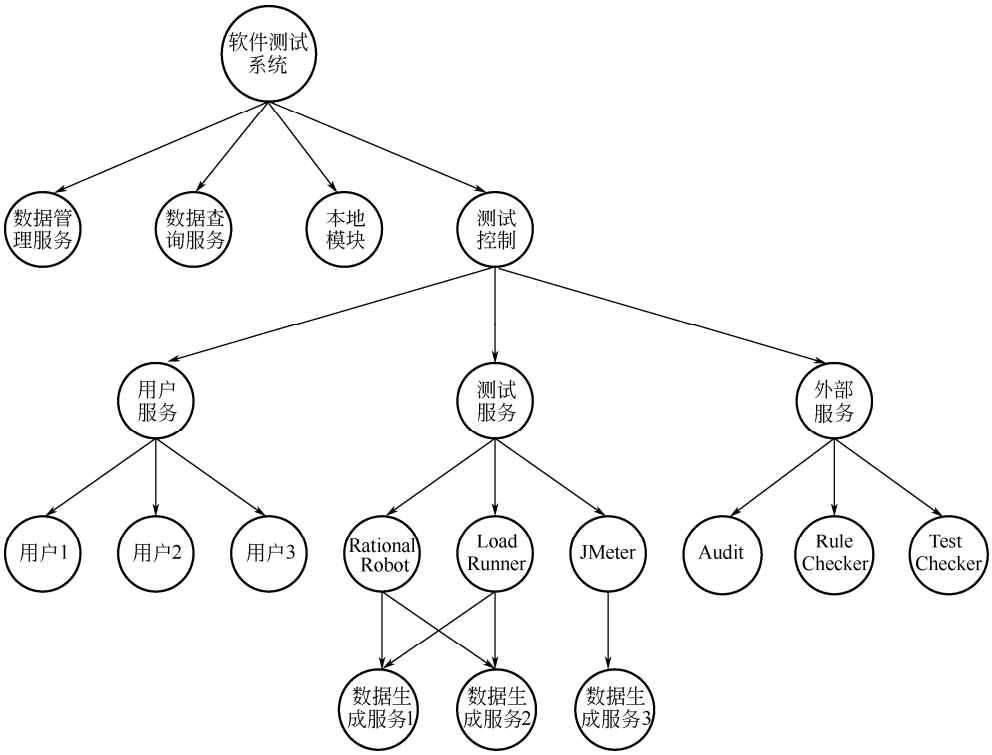


图 6.7　软件测试系统结构相依性图

6.3.2　系统可信性计算

采用自底向上的方法逐层计算系统的可信性，首先计算最下层节点的可信性。单个节点的可信性是由其先验指标汇总后经过评估指标修正

而得的，即计算先验概率并通过事实证据进行修正。以实体数据生成服务 1 为例，获取到的先验指标为功能性 0.98、可靠性 0.96、易用性 0.96、效率 0.95、可维护性 0.92、可移植性 0.90、可复用性 0.94。通过评估其开发及管理情况获得事实证据对先验概率进行修正，建立如表 6.1 所示的条件概率表。

表 6.1　评估指标条件概率表

良好测试	规范编程	详细文档	熟练员工	过程管理	可信性
1	1	1	1	1	1
1	1	1	0	1	1
0	0	0	1	1	0
1	1	1	1	0	0
1	1	1	1	0	1
0	1	1	0	1	0
1	0	0	1	1	0

表 6.1 显示了对软件开发中各条件的不同满足情况而带来的可信性差异，也即不同条件下使软件可信的概率。表 6.1 中值为 1 的项表示满足该情况，值为 0 的项表示不满足。例如，第一条数据表示软件在具有良好测试、规范编程、详细文档、熟练员工、过程管理的情况下，其可信性可以得到保证。此表通过汇总软件开发过程中出现的各种情况而得，也可在软件使用过程中根据软件的具体演化过程逐步修改完善。该表中没有列出全部数据，仅列出了几条代表性数据。

根据式（6.3）对实体数据生成服务 1 的可信性进行计算，为方便描述，将式（6.3）再次给出如下：

$$P(D|M)=\frac{P(M_e|D)P(M_p)}{P(M_e)} \tag{6.7}$$

等式左边即为修正后的节点可信性。等式右边 $P(M_e)$ 为评估指标满足给定条件的联合分布概率；$P(M_e|D)$ 为软件可信条件下各评估指标满足给定条件的联合分布概率；$P(M_p)$ 为各先验指标综合所得的可信性值，由于各先验指标相互独立，所以

$$P(M_p)=\prod_i P(M_{pi})$$

M_{pi} 分别为前文所列出的 7 种先验指标。此处根据评估指标满足情况的不同分别进行计算，后面的结果分析中将会对各计算结果进行具体讨论。

情况 1：前文所述的 5 种评估指标都满足。根据评估指标条件概率表计算 5 种评估指标联合概率可得 $P(M_e)=0.33$，软件可信条件下满足 5 种评估指标联合概率 $P(M_e|D)=0.24$，将数据代入式（6.7）可得

$$P(D|M)=\frac{0.33\times(0.98\times0.96\times0.96\times0.95\times0.92\times0.90\times0.94)}{0.24}=0.918$$

情况 2：前文所述的 5 种评估指标中满足良好测试、规范编程、详细文档、过程管理。根据评估指标条件概率表计算 5 种评估指标联合概率可得 $P(M_e)=0.39$，软件可信条件下满足 5 种评估指标联合概率 $P(M_e|D)=0.31$，将数据代入式（6.7）可得

$$P(D|M)=\frac{0.39\times(0.98\times0.96\times0.96\times0.95\times0.92\times0.90\times0.94)}{0.31}=0.84$$

情况 3：前文所述的 5 种评估指标中满足良好测试、规范编程、熟练员工。根据评估指标条件概率表计算 5 种评估指标联合概率可得

$P(M_e)=0.43$，软件可信条件下满足 5 种评估指标联合概率 $P(M_e \mid D)=0.37$，将数据代入式（6.7）可得

$$P(D \mid M)=\frac{0.43\times(0.98\times0.96\times0.96\times0.95\times0.92\times0.90\times0.94)}{0.37}=0.776$$

情况 4：前文所述的 5 种评估指标都不满足。根据评估指标条件概率表计算 5 种评估指标联合概率可得 $P(M_e)=0.09$，软件可信条件下满足 5 种评估指标联合概率 $P(M_e \mid D)=0.13$，将数据代入式（6.7）可得

$$P(D \mid M)=\frac{0.09\times(0.98\times0.96\times0.96\times0.95\times0.92\times0.90\times0.94)}{0.13}=0.462$$

依上述方法完成全部基础实体可信性的计算后，开始进行中间节点可信性的计算。计算中间节点可信性时，除要考虑该节点自身的实体可信性外，还应通过其基础节点的可信性进行修正。以实体 Load Runner 为例，计算得出该节点可信性为 0.94，其子节点的可信性分别为 0.918 和 0.928，则修正后的实体可信性为

$$P(D \mid M)\bullet P(D \mid D')=0.94\times0.928=0.87$$

其中 D' 为其基础节点数据生成服务 1 与数据生成服务 2 中较高的可信性值，$P(D \mid D')$ 也即基础节点可信时当前计算节点的可信性。系统投入运行前 $P(D \mid D')$ 的值可设为某一初值，如 $P(D')$；当系统运行一段时间后，可根据具体运行情况统计出证据数据进行修正。

另外，当一个节点可同时使用多个基础节点时，通常采用冗余备份的方式同时与多个基础节点建立连接，当其中一个失效时，立即切换至备用节点，可大大增强整个系统的可信性。此时该节点的可信性采用所有基础节点都失效的概率进行修正：

$$P(D|M)\bullet(1-\prod_i(1-P(D|D_i)))=0.94\times(1-(1-0.918)\times(1-0.928))=0.934$$

按照此方法沿结构相依性图自底向上进行计算，最终得到整个系统的可信性为 0.909。当选择的实体不同，或其满足的评估指标不同时，计算整个系统可信性所得的结果也不相同。

6.3.3 结果分析

对前述计算过程可进行如下分析：

（1）评估指标满足情况的差异性。在单个节点可信性计算的过程中，节点对评估指标的满足情况不同时，根据条件概率表计算所得的联合分布概率也不同。节点所满足的评估指标越多，即节点实体设计、开发、管理过程越符合标准规范，其软件质量水平也就越高，则实体的可信性也就越高，完全符合软件质量保证理论的思想。而计算结果也正确体现了这一点，对于先验指标完全相同的实体，根据其评估指标满足情况的不同，所得修正后的可信性也不同。也就是说，对评估指标满足程度越高的实体修正后的可信性值越高；反之，对评估指标满足程度越低的实体修正后的可信性值越低。

（2）节点间可信性的修正效果。中间节点可信性除根据该节点自身的先验指标和评估指标计算外，还受到其基础节点可信性的影响。这反映出真实情况下网构软件实体间的相互关系及一组相互协作的实体所具有的集成可信性。而通过节点冗余备份的方法，减少实体停止服务的可能，提高实体的可信性，也完全符合网构软件系统设计与开发的理论常识。

（3）最终可信性值的客观性。最终的可信性值将多个对实体可信性不同侧面的评价综合成唯一的结果，并加以修正。这样可以清晰地反映网构软件实体的可信性，同时也不单独依赖于服务提供商所给出的数据，而是可由用户根据服务实体的实际情况做出客观评估。

综上所述，本书的计算方法涵盖了从网构软件实体到系统整体的可信性计算过程，反映出网构软件各实体开发过程规范性不同所造成的可信性差异，以及网构软件系统的自主性、协同性、多态性等特征，能够正确、客观、全面地评价系统及其各组成实体的可信性。

第 7 章

网构软件动态可信性指标评估技术

当前计算机软件和互联网技术的发展，使基于 Web 的应用软件得到了日益广泛的应用。越来越多的网构软件的关键业务要基于 Web 应用完成，评估 Web 应用可信性已成为进行网构软件动态可信性指标评估的主要组成部分。目前，对 Web 应用可信性的最大威胁来自注入攻击、跨站脚本、失效的认证和会话管理、不安全的对象直接引用、跨站请求伪造等利用系统漏洞的网络攻击行为。因此，在部署和使用 Web 应用时的系统安全漏洞测试是进行动态可信性指标评估的主要手段。本章针对 SQL 注入这一最主要的系统安全漏洞攻击，从优化 SQL 注入安全漏洞渗透测试用例的角度，研究提高渗透测试准确度的问题，提出基于形式化模型的渗透测试用例。通过建立形式化模型和覆盖准则的方法指导对 SQL 注入渗透测试使用什么用例、使用多少用例等问题。研究生成优化用例集结构，达到全面测试 Web 应用黑名单过滤防御，减少对 SQL 注入漏洞漏报的目的。

7.1　攻击模型驱动的 SQL 注入渗透测试框架

本节首先剖析 SQL 注入攻击的各种手段，并分析目前 Web 应用 SQL 注入安全漏洞渗透测试的方法和其相关研究，指出目前相关研究在 SQL 注入渗透测试用例方面所存在的无规律、易导致漏报等方面的问题。在此基础上，指出对目前 SQL 注入随机枚举用例方式进行改进的必要性并提出 SQL 注入优化用例集合的标准。为了实现生成优化 SQL 注入渗透测试用例的研究目的，本节提出一种攻击模型驱动的 Web 应用 SQL 注入渗透测试框架方法。测试框架的主要思想是通过对实际 SQL 注入的各种攻击手段进行形式化建模，以形式化攻击模型表达 SQL 注入的攻击位置、攻击输入和 SQL 注入安全漏洞反应特征方面的规律，进而以这些规律信息指导渗透测试过程，表述所使用的 SQL 注入用例应具备的规律，以达到指导生成优化 SQL 注入渗透测试用例集合的目的，同时以此渗透测试框架作为研究的展开思路和整体结构框架。

7.1.1　SQL 注入攻击与安全漏洞种类剖析

SQL 注入攻击本质上是一种通过操纵输入来修改软件后台 SQL 语句以达到利用代码进行攻击目的的技术。具体的 SQL 注入实现手段有

很多种，通过对当前揭露的各种 SQL 注入攻击方式[200]的分析，根据目前相关研究中[206,210]经常采用的分类方法，进一步详细归纳 SQL 注入的各种攻击手段及造成的安全漏洞如下。

1．重言式攻击（Tautology）

重言式攻击是一种简单有效和常用的攻击方式，重言式攻击的基本目的是对系统原 SQL 命令语句的条件子句插入一个重言式（永真式），使得 SQL 命令语句条件子句部分恒成立，即使原 SQL 语句条件子句的条件限定失效。这种攻击的效果与途径视具体 Web 应用而定，但一般应用于绕过认证机制和窃取系统信息方面。这种形式的攻击中，攻击者往往找到用于 SQL 命令语句 where 条件语句的输入点来输入重言式。SQL 语句的条件子句用于搜索后台数据库中每条符合条件的记录，若攻击者将条件子句改为了永真式，则 SQL 语句会匹配返回数据表中的所有记录。

2．联合查询（Union Queries）

尽管重言式攻击可以在绕过认证等场合下达到一定的攻击目的，但对于窃取特定信息等攻击目的其显得灵活性不足。而联合查询攻击则是一种更为精巧的攻击形式，可以达到此类目标。联合查询攻击可以通过额外插入的“合法的”查询语句获得额外信息。在这种攻击中，攻击者输入“UNION <查询语句>”的形式作为攻击输入，以使插入的查询语句获得执行来窃取特定数据表中的信息。这种攻击的输出是 UNION 之后攻击输入查询语句所执行而获得的数据集合。

3．多命令语句攻击（Piggybacked Queries）

与联合查询攻击相似，多命令语句攻击也是通过输入额外的可执行命令语句作为攻击输入。区别在于联合查询攻击只能输入形如

“UNION SELECT”的攻击语句，而多命令语句攻击不使用“UNION”关键字作为命令之间的分隔符，因而不局限于“SELECT”这一种命令动词构造的命令语句，故可以输入更多样的 SQL 命令动词所组成的命令语句。若此种攻击成功，后台数据库会执行所插入的一条或多条命令语句，达到攻击者的相应目的。这种攻击方式危害较大，因为攻击者实际上可以插入任意的 SQL 命令语句使其运行来实现任意的攻击目的。

4．异常命令语句攻击（Malformed Queries）

联合查询攻击和多命令语句攻击使攻击者可以插入并执行命令语句达到自己的攻击目的。而异常命令语句攻击通过构造畸形的 SQL 命令语句作为攻击输入，这些畸形语句会导致系统返回异常出错信息，攻击者通过那些过分具有提示性的异常出错信息，获得系统内部的相应信息。例如，推断出后台数据库类型、数据表结构等信息。攻击者往往通过输入语法错误的 SQL 命令语句或异常字符诱发系统返回出错信息以达到攻击目的。

5．推断攻击（Inference）

推断攻击也是一种窃取系统信息的攻击手段，其通过构造输入 SQL 查询条件，使得系统对这些查询条件成立与否的不同状态产生不同的反应，通过观察系统不同反应，获知系统内部信息。这种攻击手段使攻击者可以在系统不产生返回出错信息或命令语句执行结果的条件下，实现窃取信息的攻击目的。此种攻击的一种典型手段是时间推断（timing attack），攻击者通过输入包含条件式的时间推断命令，观察后台数据库是否反应延时等情况，得知条件式成立与否，以此推断获得系统内部信息。另一种典型的推断攻击方式是条件猜解攻击（conditional inference

或 blind injection），这种攻击通过输入一系列为真及为假状态不同的条件式，使系统对真假条件式产生不同的反应，使攻击者观察得到系统内部信息。

6．攻击输入编码伪装（Alternate Encodings）

上述 SQL 注入攻击需要使用一些特定的 SQL 关键字或特殊字符作为攻击输入，如单引号、分号、SQL 命令动词等。因此一些简单的 SQL 注入攻击防御机制采用检查用户输入是否包含有这些特殊的关键字来阻止 SQL 注入攻击输入。所以为了应对这一类基于关键字过滤的防御措施，产生了攻击输入编码伪装的 SQL 注入攻击方式。编码伪装使攻击者可以改变攻击输入的外在样式，以躲避和突破关键字过滤式防御机制，实现成功攻击。对攻击输入的编码伪装方法一般有 ASCII 编码、十六进制编码以及 Unicode 编码[224]等，这些方法可使攻击输入转化为与原先全然不同的样式，同时保持相同的含义和作用，以此来欺骗仅过滤非编码 SQL 关键字符的防御机制。在一些情况下，即使防御措施考虑到了对攻击输入编码伪装的过滤，此种攻击也有成功的可能，因为其可以对 Web 应用不同的层次进行攻击。比如，防御机制可能会对 Unicode 或十六进制编码攻击输入进行过滤，但攻击者可以使用数据库函数对攻击输入进行编码。这就表明对编码伪装攻击方式绝对有效的防御，必须在所有应用层上采用可防御过滤所有编码伪装方式的机制，这在实际中是很难做到的。

7．存储过程攻击（Leveraging Stored Procedures）

一项被认为对 SQL 注入攻击十分有效的防御措施就是使用存储过程。存储过程增强了数据库业务处理规范性并独立于具体的 Web 应用逻辑，相当于为系统提供了额外的抽象层。然而，认为仅仅使用存储过

程就可以杜绝 SQL 注入攻击是错误的。实际上与其他的软件系统类似，存储过程本身的安全性也取决于其自身具体编程实现的方式以及对攻击防御的充分性。因此参数化的存储过程依然会像 Web 应用其他部分的代码一样，存在被 SQL 注入成功攻击的问题。

通过以上对 SQL 注入各种攻击手段进行的分类剖析，我们可以得出以下三点重要结论。

（1）SQL 注入攻击本身具有极为丰富和复杂的攻击输入方式，因此对于彻底充分的 Web 应用 SQL 注入安全漏洞渗透测试来讲，应使测试用例全面地反映各种 SQL 注入攻击手段和攻击输入样式，以确定 Web 应用防御机制的充分性，准确判定 SQL 注入安全漏洞存在性。

（2）SQL 注入的攻击输入实际上是一个无限集合，因此虽然如上分析 SQL 注入用例应当包括各种攻击方式和攻击输入样式，但以目前相关研究中所采用的随机枚举的用例方式是不可能做到的。所以必须采用某种形式化表达方式有规律地描述 SQL 注入用例，在确保用例反映多种 SQL 注入攻击种类以及样式的同时，实现对用例描述的规律性和有限性。

（3）SQL 注入攻击输入和安全漏洞特征自身有规律可循。通过上述对 SQL 注入攻击手段的剖析，可以看出尽管具体的 SQL 注入攻击输入样式纷繁复杂，总体数量无穷多，但 SQL 注入的每个攻击输入都处在一个特定的攻击类别当中，具有一个特定的攻击目的。因此，这样的事实为我们基于 SQL 注入攻击目的，采用形式化方法建立描述 SQL 注入攻击规律的模型提供了现实可能性。

7.1.2 攻击模型驱动的渗透测试框架

基于以上分析，为了实现提高 SQL 注入渗透测试准确度的根本目的，应生成优化的 SQL 注入用例集合，而相对于目前随机枚举的 SQL 注入用例，这个优化的用例集合应至少具备以下特征。

（1）用例集合所包含的用例输入能够充分反映当前各类 SQL 注入攻击方式，以充分测试 Web 应用在受到各种 SQL 注入手段攻击时是否存在安全漏洞。

（2）用例集合所包含的用例输入具备各种 SQL 注入攻击输入样式，以测试 Web 应用是否可防御各种 SQL 注入攻击输入。

（3）用例集合的生成在一定的规则指导下进行，从而实现测试知识的共享和可重用性。

（4）对用例集合所包含用例输入的表述规则化、有条理，且用例集合所包含用例输入的充分性（覆盖度）是可衡量的。

（5）用例集合应能体现攻击输入、漏洞反应特征和攻击输入位置等方面的信息、使渗透测试过程实现对 SQL 安全漏洞存在性的准确判定。

（6）所生成的用例集合包含的用例输入数量是有限的，且此用例集合实际运行的时间、空间等代价在可接受范围之内。

以上所述优化用例集合的特征，基本上是当前随机枚举 SQL 注入用例方式所不具备的。因此为达成生成优化 SQL 注入用例集合的研究目标，本书提出了一种攻击模型驱动的渗透测试框架（图 7.1），该测试框架通过对 SQL 注入攻击建模来描述 SQL 注入各攻击方式的规律性，

进而指导生成优化的 SQL 注入渗透测试用例集合。

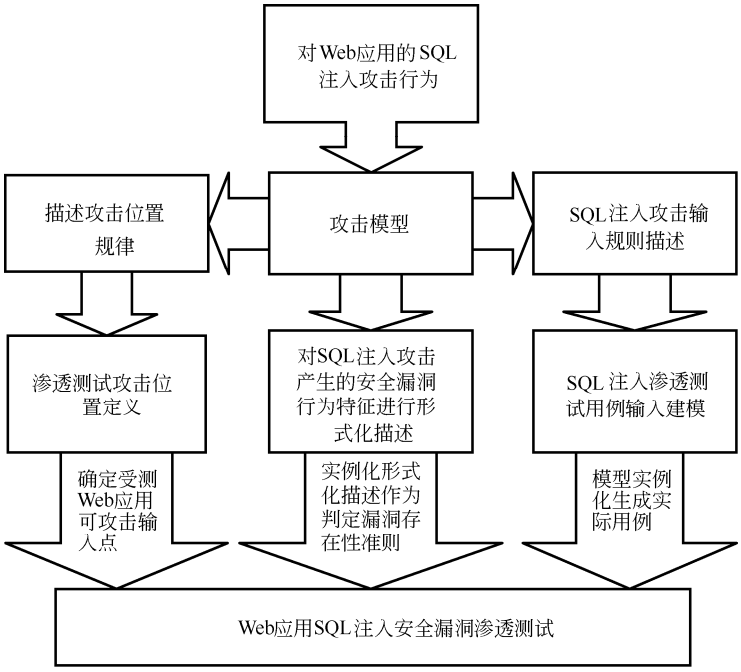


图 7.1　攻击模型驱动的渗透测试框架

所提出的渗透测试框架，针对上述生成优化用例的研究问题，以框架中的 SQL 注入攻击模型描述各种 SQL 注入攻击手段的规律，以实现指导渗透测试过程中对其各种攻击手段的较全面展现和考虑。如图 7.1 所示，SQL 注入攻击模型描述的主要规律信息和指导包括：以攻击模型描述的 SQL 注入攻击输入位置作为用例可输入位置指导信息（左路），以对 SQL 注入安全漏洞行为特征的形式化描述作为较准确安全漏洞存在性判定规则（中路），以框架中攻击模型描述的 SQL 注入攻击输入规则指导生成优化的渗透测试用例输入（右路）。本研究通过一系列的形式化模型及对其进行的实例化，最终生成符合上述优化特征的 SQL 注入测试用例。

当前相关研究中 SQL 注入用例之所以采用随机枚举的方式，主要原因是对当前各种 SQL 注入手段的内在规律缺乏有效抽象描述，对各种具体的SQL注入攻击输入也缺乏除枚举之外的有条理、有规律的表述方法。而这正需要对其建立形式化模型的方式加以解决。形式化建模是生成有条理 SQL 注入用例的必要手段。如同对 Web 应用基于模型功能测试的研究[198]中，往往要考察对 Web 应用的用户使用行为，通过对使用行为进行建模，以模型化有条理地表述使用行为的规律来驱动完善的 Web 功能测试；同样对于 Web 安全测试的渗透测试，也应考察对 Web 的攻击行为，以对攻击行为建模的方案驱动渗透测试实施，这样的研究思想是相互对应的。这也是我们提出上述渗透测试框架的必要性。

所提出的渗透测试框架，属于测试模型的研究领域。目前在软件测试模型的研究方面，一些典型的测试模型包括基于模型检查的测试模型[211]、基于 EAI 的软件测试模型[207]、基于构件的软件测试模型[212]等，相关研究所提出的软件测试模型数量和种类很多。但是总体来看，在目前已经提出的众多的软件测试模型中，大多数用于软件性能测试和功能测试，或者针对一般软件的漏洞测试，适用于发现 Web 应用安全漏洞的软件安全测试模型不多[205]。测试模型大多仅针对软件的性能和功能测试问题，而少有考虑安全性测试问题。特别是对于本书所定义的 SQL 注入用例问题不能提供很好的专门性指导。因此，本章提出上述渗透测试框架来为 Web 应用安全漏洞渗透测试及所研究的用例问题提供指导框架。

为实现上述提出的渗透测试框架模型，后续研究工作将围绕框架描述内容展开，分别对 SQL 注入攻击手段建模、安全漏洞行为特征形式化定义和 SQL 注入攻击输入规则描述等几方面的问题开展研究。这些

研究内容是对图 7.1 中渗透测试框架模型的细化。

7.2　基于安全目的模型的 SQL 注入攻击建模

为了实现本章所提出的攻击模型驱动的渗透测试框架，本节中探讨对 Web 应用 SQL 注入攻击行为的建模。在本章所提出的渗透测试框架中，攻击模型是测试框架的一个核心部分，其将实际的 SQL 注入攻击行为转化为模型化描述，为渗透测试提供攻击位置、攻击输入、Web 应用漏洞反应规律方面的信息，最终生成所述的优化 SQL 注入用例指导信息。因此，对 SQL 注入建模问题的研究对优化 SQL 注入用例问题具有重要作用。

7.2.1　渗透测试中攻击建模的必要性

渗透测试的特点是模拟攻击式安全测试[201]。Web 应用安全测试中的渗透测试最初由黑客技术发展而来，其基本思想是通过对 Web 应用的模拟攻击，测试受测系统的脆弱性。基于这种测试思想，在脆弱性评估技术发展的初期，主要通过罗列实际应用环境中已知的允许的攻击手法作为渗透测试用例，模拟攻击以测试系统的脆弱性。具体到 SQL 注入渗透测试，这就是前文所述的目前相关研究中的随机枚举 SQL 注入用例方式。这种方式实际就是一个不断积累实际攻击方式经验、提取攻

击成功规则到规则匹配判定漏洞存在性的过程。所以其重点技术就是如何产生更加准确和完备的攻击规则集，可以说目前广泛应用的 Web 应用安全漏洞扫描工具[202~204]就是这种思想的体现。

而这种随机枚举 SQL 注入用例方式除了会带来漏报等问题而影响测试准确度之外，还会带来其他问题。例如，渗透测试的用例集合与杀毒软件的病毒库类似，故这种方式存在与病毒库更新机制相似的固有缺点：需要保证攻击行为的不断更新机制，以罗列产生攻击规则集。而这产生了与病毒库检测病毒机制一样的问题：实际攻击的发生总是早于防御/检测的手段归纳。即先发现一个特定的攻击手段，再加入测试用例集合，而不是通过已经发现的攻击手段举一反三，提炼其中的规律来形成前瞻性的用例集合。这容易导致模拟攻击安全检测的滞后。而且，不断罗列的攻击规则也造成了日益庞大和冗余的测试用例，繁杂无规则的攻击参数/方法集合给模拟攻击安全测试的实际实施带来了困难。

因此对生成优化 SQL 注入用例集合攻击手段的研究中，为了实现对攻击手段的全面性、规律性描述，客观上要求使用形式化方法描述攻击知识集合、以模型化的方法表示模拟攻击模式。因为应用形式化的建模描述方法，才有可能从整体上全面描述攻击手段的内在规律，将具体的攻击规则集抽象成为逻辑层次的形式化描述，比规则罗列的方式更能表达其本质。例如，对于 Web 应用的安全测试，可以通过使用攻击建模研究领域中的建模方法，基于故障树、有限状态机、Petri 网、攻击图等工具对 Web 应用的攻击行为建模，攻击模型的建立可为渗透测试这种模拟攻击式安全测试提供更加全面的对攻击行为的理解，有助于指导建立更有效的漏洞模拟攻击用例。

一些相关研究中提出了基于模型的 Web 应用测试方法[198]。这些研

究方法的核心思想是，将用户对 Web 应用的使用行为（或 Web 应用结构）抽象建模，从而将直观上无法观察到内在规律的复杂的 Web 应用使用行为集合，以模型的形式表述出来[213~220]，以此实现用模型表述的 Web 应用行为规律信息来驱动对 Web 进行较充分的功能测试[218]，这比直接罗列并枚举使用无规律的用户使用行为集合能够更全面地测试 Web 应用功能是否完善（对 Web 应用结构建模的思想与此类似）。而如同用户的使用模型之于 Web 功能测试的作用一样，攻击模型对 Web 渗透测试的重要性在于，其以抽象方式形式化表达系统受到的攻击威胁，描述攻击行为的全貌和逻辑层面，提高渗透测试者考虑攻击行为的全面性、掌握攻击行为的规律性。因此为了实现目前 SQL 注入用例无规律枚举向有规律描述和使用的发展，有必要将攻击建模和渗透测试这两个研究领域相结合，建立 SQL 注入攻击模型来指导渗透测试，这对于实现上述以测试框架指导生成优化 SQL 注入用例有重要作用。

7.2.2　基于攻击者直接目的的 SQL 注入攻击树模型

为了解决目前对于 SQL 注入攻击建模的不足，进一步完善 SQL 注入攻击建模，本书提出了基于攻击者直接目的的分析方法，对 SQL 注入各种攻击方式和攻击输入规律进行分类和总结，在此基础上建立更加完善的 SQL 注入攻击树模型。

在 SQL 注入的目的方面，文献[210]中总结了十大类 SQL 注入攻击目的，分别是识别可注入位置（Identifying Injectable Parameters）、识别后台数据库特征（Performing Database Finger-Printing）、确定数据库结构（Determining Database Schema）、窃取数据（Extracting Data）、篡改数据

（Adding or Modifying Data）、拒绝服务攻击（Performing Denial of Service）、逃避探测（Evading Detection）、绕过认证（Bypassing Authentication）、注入运行恶意命令（Executing Remote Commands）、权限提升攻击（Performing Privilege Escalation）。

在此基础上，本书在全面研究了当前 SQL 注入攻击的行为之后，从更加普遍的意义上，将 SQL 注入按照攻击者的直接目的（immediate purpose），进一步简化为三大类：恶意运行命令、绕过认证、窃取系统信息。简化的目的是使建立的攻击树模型更加简明紧凑，减少冗余。在此，攻击者直接目的是指 SQL 注入攻击输入向 Web 应用提交后最先一步所要实现的目的。根据这种思路，攻击者直接目的与一般攻击目的的对应如表 7.1 所示。

表 7.1　攻击者直接目的与一般攻击目的的对应

	窃取系统信息	绕过认证	恶意运行命令
识别可注入位置	√		
识别后台数据库特征	√		
确定数据库结构	√		
窃取数据	√		
篡改数据			√
拒绝服务攻击			√
逃避探测		√	
绕过认证		√	
注入运行恶意命令			√
权限提升攻击			√

基于 SQL 注入攻击者直接目的，通过对当前 SQL 注入攻击手段的

全面考察，建立攻击树模型，如图 7.2 所示。

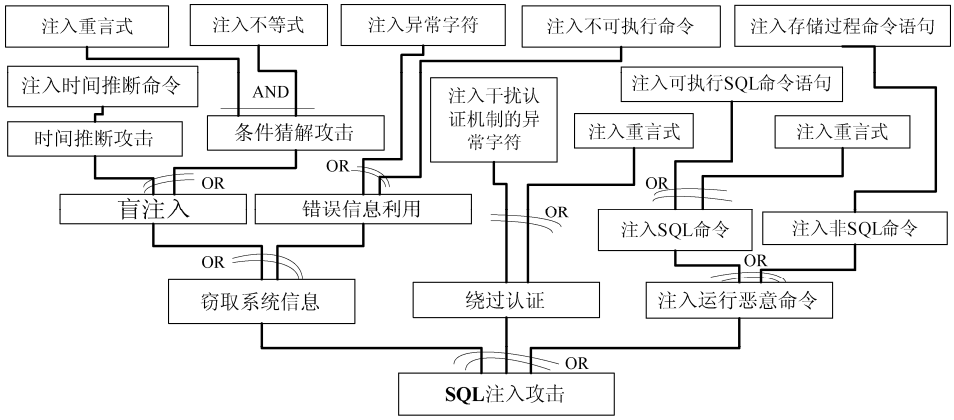


图 7.2　基于攻击者直接目的的 SQL 注入攻击树模型

对比文献[208]及[209]和图 7.2 不同的 SQL 注入攻击模型，可看出对 Web 应用安全漏洞攻击的建模，至少有三种思路：基于攻击手段的建模[208]、基于攻击流程的建模[209]和基于攻击目的的建模（图 7.2）。本书所提出的模型，从攻击者直接目的的角度将 SQL 注入攻击分为窃取系统信息、绕过认证和注入运行恶意命令三大类，进而在模型中自底向上分别描述实现这三类攻击目的的手段。例如，对于窃取系统信息的攻击目的，可由盲注入或错误信息利用的 SQL 注入攻击方式实现；绕过认证的攻击目的可由注入异常字符或重言式攻击实现等。

如图 7.2 所示，本书所提出的攻击树模型将 SQL 注入攻击描述为盲注入、错误信息利用、绕过认证、注入 SQL 命令和注入非 SQL 命令几类，其与一般 SQL 注入攻击分类的对应如表 7.2 所示。文献[208]中所提出的 SQL 注入模型对于盲注入、注入存储过程等种类的攻击未加描述，而本书所提出的模型可全面当前各类 SQL 注入攻击[206]，且模型的描述不着眼于具体的攻击程序语句格式[208]（attack signature）或对

目标系统具体的攻击步骤[209]，这种描述角度有利于支持将攻击输入代表符号与其具体样式的描述分离，从而实现对原本无序的 SQL 注入攻击输入集合，进行有序的形式化符号表达。与描述具体攻击流程或具体攻击输入的建模角度相比，从攻击目的的角度建模使攻击模型更不易过时和抽象层次高。例如，文献[208]中提出的攻击树模型以正则表达式描述当前 SQL 注入攻击输入，当具体攻击输入手段发展变化时该描述即会过时；而文献[209]中的模型更要随着具体 Web 应用的变化而变化。而图 7.2 所示的模型将建模抽象到描述攻击目的层次，由于攻击目的相对于具体攻击手段、步骤是稳定的、不易变化的，所以所提出的模型具有不易过时的稳定性。

表 7.2　本书提出的攻击树模型与一般 SQL 注入攻击分类的对应

	盲注入	错误信息利用	绕过认证	注入 SQL 命令	注入非 SQL 命令
重言式攻击	√		√	√	
联合查询				√	
多命令语句攻击			√	√	
异常命令语句攻击		√			
推断攻击	√				
攻击输入编码伪装	√	√	√	√	√
存储过程攻击					√

以上提出了基于攻击目的的 SQL 注入攻击树模型，为指导生成优化渗透测试用例提供了基础。图 7.2 所示的模型主要描述了 SQL 注入的攻击输入方面的规律，因此可据此生成较有规律的 SQL 用例输入，但其对于 SQL 注入攻击位置规律和漏洞反应规律尚未进行描述，而在图

7.2 所示的 SQL 注入攻击模型中继续丰富关于攻击位置和漏洞反应等方面的信息也是可以实现的，但这会引起攻击树模型中较多的冗余和重复分支且难以保证描述一致性。这主要是由于攻击树模型自身特点和定义所限，攻击树的一些节点无法避免产生冗余和重复，这对划分实际的 SQL 注入攻击输入规则集合是不利的。正如一些研究表明[221]攻击树建模方式自身存在一些缺点，如攻击行为和结果都用节点表示，容易造成混乱；树分支和节点易产生冗余和重复等。另外，攻击树模型的表达能力不足，导致一些额外的重要信息（如探查 Web 应用输入点、攻击手段之间的信息联系等）难以被表达出来。因此尽管本 SQL 注入攻击树模型可体现当前 SQL 注入攻击输入规律，但仍需要采用性能更加完善的建模方法对 SQL 注入攻击进行描述，以描述攻击输入、漏洞反应和攻击位置方面的信息，又不会使所建立的模型本身过于冗余和繁杂。

7.2.3　基于安全目的模型的 SQL 注入攻击模型

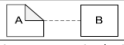
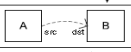
上一节出了基于攻击者直接目的的 SQL 注入攻击树模型，它比当前相关研究中所提出的 SQL 注入攻击模型可更全面地描述 SQL 注入攻击输入规律，有利于指导 SQL 用例的规律。同时分析也指出以攻击树建模在表达信息方式上易造成模型的冗余，且其对于 SQL 注入攻击位置和漏洞反应的规律不易表达。因此在上述基于攻击树的 SQL 注入模型基础上，本节提出基于安全目的模型（Security Goal Model，SGM）的 SQL 注入攻击模型，以更全面地描述 SQL 注入攻击规律，并减少模型表达上的冗余和重复。

安全目的模型（SGM）是一种用来描述漏洞、安全特性、攻击或安

全软件开发的新型模型方法[222]。其特点是表达能力较好，可用来表述攻击树、漏洞成因图、安全行为图和安全目标指示树等模型所表述的逻辑内容。SGM 可代替此 4 种建模工具，并可以通过转换规则与其他的建模工具互相转换。

安全目的模型的建模规则如表 7.3 所示。

表 7.3　安全目的模型建模规则

Symbol	Element
	Vertex representing an subgoal that is not associated with a security goal model. *A* is a contributing subgoal; *B* is a counteracting subgoal.
	Vertex representing a subgoal associated with a security goal model. *A* is a contributing subgoal; *B* is a counteracting subgoal.
	Root.
	Dependence edge with stereotype. *B* depends on *A*. The edge has stereotype *type*.
	Note annotation.
	Annotation edge; *A* is an annotation for *B*.
	Operation *or* and operation *and*, with three operands each. Text labels are optional.
	Information edge. There is an information edge connecting information port *src* of vertex *A* to information port *dst* of vertex *B*.
	Port type indications: *src* is an output port and *dst* is an input port. These indications are optional.
	Input (named *input*) and output (named *output*).

在安全目的模型中，根节点表示可由实现各子目的元素而达到的总目的，一个安全目的模型中只有一个根节点。子目的（subgoal）表示一个有助于实现（以白色框表示）或不利于实现（以黑色框表示）模型总目的的局部目的，一个安全目的模型中可以含有多个子目的。模型中的操作符（operators）包含 AND 和 OR 两种，分别表示“与”和“或”的子目的间可实现关系。安全目的模型的依赖边（dependence edge）表示子目的之间 AND 或 OR 的依赖关系，一条从 A 指向 B 的依赖边（有向箭头）表示为了实现本模型所描述的总目的（根节点），子目的 A 与 B 应当依次实现。安全目的模型还具有描述子目的之间信息流的能力，可以用信息边（information edge）表述子目的之间传递信息的情况，用

信息节点（information port）表达子目的内部对外接收和发送信息的接口状况。在安全目的模型中信息边是非必需的，当使用安全目的模型描述安全漏洞等方面的规律时信息边是可以省略的。

与攻击树相比，SGM 的主要优点在于表达简洁且较少冗余，规模不大而表达能力强。以对篡改软件（replace software）攻击的建模[222]为例，若以攻击树为工具建模，相应的攻击树模型如图 7.3 所示。而同样的此种攻击行为以 SGM 建模可表述为图 7.4。

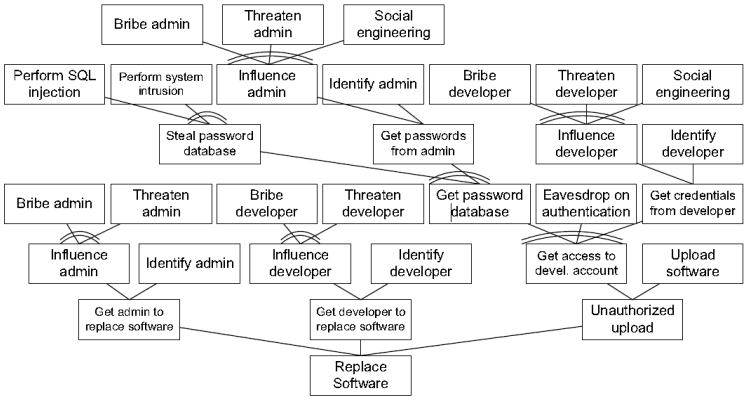


图 7.3　篡改软件攻击的攻击树模型

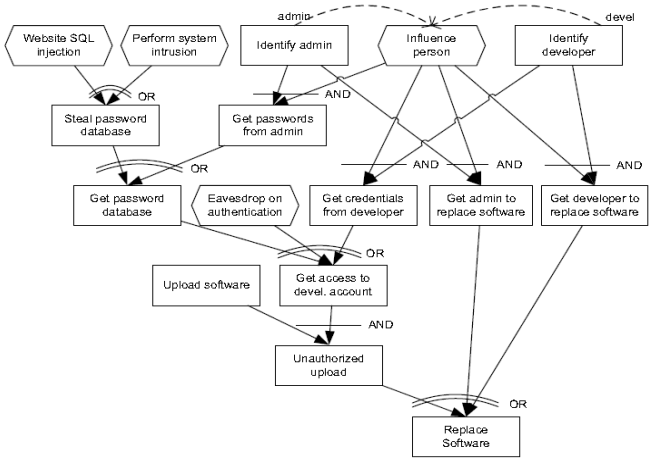


图 7.4　篡改软件攻击的 SGM 图

通过对比图 7.3 与图 7.4 可以看出，SGM 的优点在于可共享模型中相关节点，而不像攻击树模型须重复列出相同的节点（如 Influence admin 及其子节点），因此模型可以进一步简化。此外 SGM 具有更完善的表达能力，模型中可以反映更多信息[222]，具备较好的描述性。

基于安全目的模型对 SQL 注入攻击建模，首先必须挖掘 SQL 注入攻击的目的，以将 SQL 注入攻击内在规律描述向安全目的模型的子目的描述方式转化。本书所提出的基于攻击者直接目的的 SQL 注入攻击树模型正好可以满足这一要求。这体现了本书所提出的基于攻击目的建模思想的必要性。

新的基于安全目的模型所建立的 SQL 注入攻击模型如图 7.5 所示，模型首先从攻击者直接目的的角度对 SQL 注入攻击规律进行宏观描述。

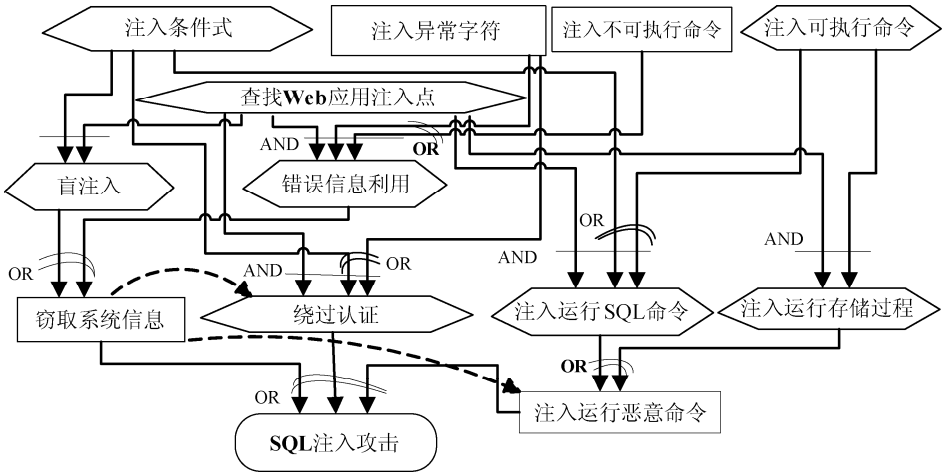
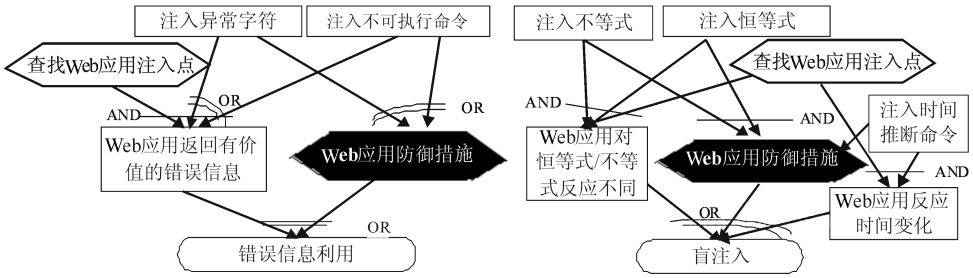


图 7.5　SQL 注入攻击模型

安全目的模型描述角度是安全相关的行为目的，基于上述攻击者直接攻击目的对 SQL 注入攻击建模。图 7.5 中模型的根节点表示实现 SQL 注入攻击的总目的，进而自底向上描述，按照攻击直接目的，将实现这

个总目的（SQL 注入攻击）的攻击分为三类子目的：窃取系统信息、绕过认证和注入运行恶意命令。模型进一步向上分别描述了实现这三种子目的各自所需的攻击子目的，如注入运行 SQL 命令需要查找 Web 应用注入点、注入条件式或注入可执行命令的子目的等，同样其他各攻击子目的的描述也是自底向上表达要实施的攻击子目的，直至模型最上端一系列的攻击注入子目的和查找 Web 应用注入点攻击子目的。

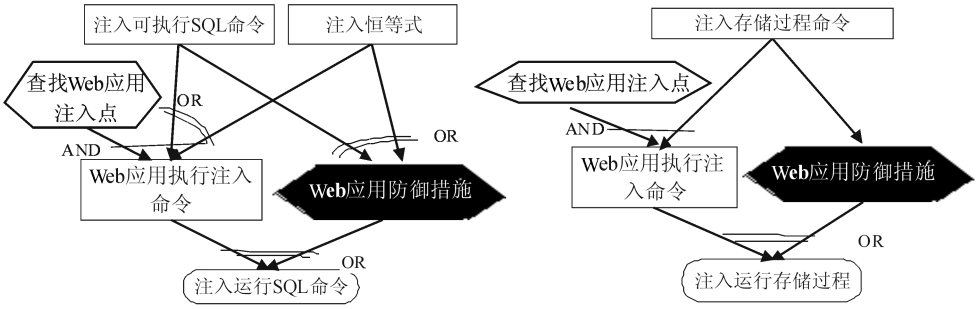
在图 7.5 对 SQL 注入规律总的描述基础上，对其中的“窃取系统信息”子目的进一步展开描述建模（见图 7.6）。在图 7.2 所示的攻击树模型中，窃取系统信息攻击分为错误信息利用和盲注入两种攻击方式。基于安全目的模型的表述，将窃取系统信息建模为图 7.6 中的两个子模型：图 7.6（a）为错误信息利用攻击子目的模型，图 7.6（b）为盲注入攻击子目的模型。其中，在错误信息利用子模型中，模型上端描述了为实现此种攻击目的要进行的攻击输入：注入异常字符或注入不可执行命令。模型中部白色方框部分描述了 Web 应用存在 SQL 注入漏洞时对攻击输入的漏洞反应：Web 应用返回有价值的错误信息；中部的黑色方框代表 Web 应用防御措施，即对 SQL 注入攻击的防御措施不利于实现成功的攻击，故以反作用（Counteracting）节点表示。在盲注入子模型中，模型上端描述了为实现此种攻击目的需要进行的攻击输入：注入恒等式与不等式或者注入时间推断命令，以分别达到模型中部白色方框部分描述的 Web 应用对恒等式与不等式反应不同以及 Web 应用反应时间变化的子目的，两者分别对应成功的 SQL 注入条件猜解攻击和时间推断攻击所具备的特征；同样模型中部黑色方框代表 Web 应用防御措施，即防御措施不利于实现成功的 SQL 注入盲注入攻击子目的。



(a) 错误信息利用攻击　　(b) 盲注入攻击

图 7.6　SQL 注入攻击中“窃取系统信息”SGM 图

同样对图 7.5 中的“注入运行恶意命令”子目的进行建模，将此子目的分为图 7.7 中的注入运行 SQL 命令攻击和注入运行存储过程攻击两个模型。注入运行 SQL 命令攻击子目的模型上端描述了两种注入攻击的形式：注入可执行 SQL 命令或注入恒等式，模型中部白色方框描述成功的注入运行 SQL 命令攻击的特征。同样注入运行存储过程攻击子模型上端描述了攻击输入，中部白色方框描述成功的攻击所具有的特征，两个模型中部黑色方框代表 Web 应用防御措施。



(a) 注入运行SQL命令攻击　　(b) 注入运行存储过程攻击

图 7.7　SQL 注入攻击中“注入运行恶意命令”SGM 图

对图 7.5 中的“绕过认证”攻击子目的展开建模，如图 7.8 所示。模型上端描述了攻击输入（注入恒等式或注入干扰认证机制的异常字

符），中部白色方框描述成功的攻击特征：通过 Web 应用认证机制，模型中部黑色方框代表 Web 应用防御措施。

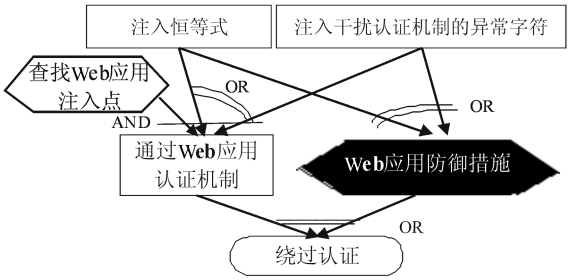


图 7.8　SQL 注入攻击中“绕过认证”SGM 图

以下进一步对图 7.6～图 7.8 中各子模型的查找 Web 应用注入点子目的建模。本书的重点是 SQL 注入用例优化问题，而不是查找注入点彻底性问题，因此只考虑 Web 应用中带参数 URL 与登录认证表单[200]两类输入点作为测试对象。在此基础上，对查找 Web 应用注入点子目的建模，如图 7.9 所示。

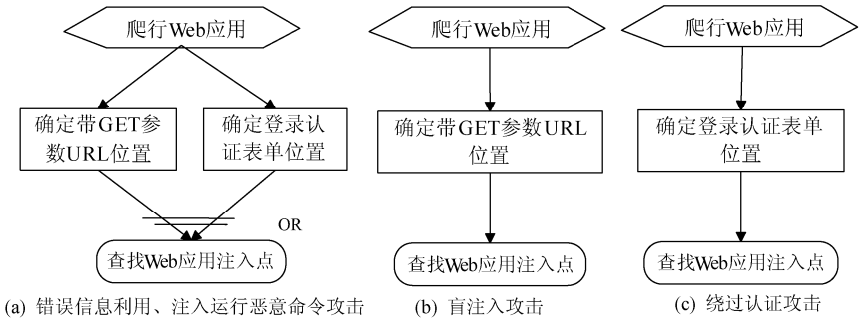


图 7.9　SQL 注入攻击中“查找 Web 应用注入点”SGM 图

图 7.6～图 7.9 中的模型是对图 7.5 所示模型的详细描述：将图 7.5 所示模型中相应的子目的展开为若干子模型，同时加入对 SQL 注入攻击位置信息和漏洞反应规律的描述。这体现了安全目的模型的优点：

可以进一步对模型节点进行分解分层描述，并支持对反作用的描述。图 7.6～图 7.8 中各模型中自上而下每条不含反作用目的节点的路径代表一种攻击子目的的过程：最上端描述了攻击输入；中部子目的节点描述了 Web 对攻击输入的漏洞反应，即 Web 有何种反应即为具有 SQL 注入安全漏洞。模型中的反作用节点（黑色）表示攻击被 Web 应用防御拦截造成不成功的攻击路径。

以上在图 7.2 的攻击树模型基础上，进一步基于安全目的模型（SGM）建立了新的 SQL 注入攻击模型，相比目前一般的攻击树建模方法，本节所提出的基于 SGM 方法对于 SQL 注入攻击的建模描述主要有以下优点。

（1）减少模型表述上的冗余，并有利于保障描述信息上的一致性，如在图 7.5 中 SGM 不必像攻击树模型那样将相同内容（如查找注入点）在各分支重复出现，减少了模型分支重复。

（2）支持对子目的展开分层描述，这项特性使模型既可以对攻击规律进行宏观描述（图 7.5），又可以对攻击规律进行细节展现（图 7.6～图 7.9），并能以模型群的形式表达各种 SQL 注入攻击，表述不同种类的 SQL 注入攻击的特点（如图 7.6～图 7.8 中对攻击输入的反应等）。

（3）SGM 支持对 SQL 注入攻击漏洞反应的描述，据此可指导生成准确的 SQL 注入输出，有助于完善 SQL 注入用例建模信息，而目前相关研究所提出的攻击树[208,209]并未考虑明确表述攻击效果、描述 SQL 注入漏洞反应特征方面的信息，因此难以充分指导生成包含预期输出信息的 SQL 注入用例集合。

（4）基于 SGM 的 SQL 注入攻击模型可以通过对子目的节点的分展描述，表达 SQL 注入攻击输入之间的分类信息。例如，在图 7.5 中描述

盲注入攻击、注入运行 SQL 命令、绕过认证三个攻击子目的都需要注入条件式，在图 7.6 中盲注入攻击模型的顶层表述注入恒等式与不等式或时间推断命令、图 7.7 中注入运行 SQL 命令和图 7.8 中绕过认证的模型顶层描述这三种子目的均须注入恒等式，以此细化表明恒等式与不等式注入及时间推断命令都属于注入条件式这一大类。在图 7.5 中注入可执行命令子目的也在注入运行 SQL 命令和注入运行存储过程两个攻击子目的中用到，图 7.7（a）和（b）分别详述了这两类攻击子目的各自所用到的攻击输入子目的：可执行 SQL 命令与存储过程命令，表明这两个攻击输入子目的都属于可执行命令这一大类。同样，图 7.6 中的注入异常字符和图 7.8 中的注入干扰认证机制的异常字符子目的，都属于图 7.5 中的异常字符一大类。对不同攻击输入的分类归属信息进行描述，为 SQL 注入用例建模等工作提供了必要的指导信息。而攻击树的叶节点之间不支持合并分展描述方式，因此难以表述其所代表的攻击输入分类信息。当前相关工作中所提出的 SQL 注入攻击模型[208]也并未对攻击输入之间的分类问题进行探讨。

（5）SGM 可对 SQL 注入攻击的 Web 应用注入点信息规律进行描述（图 7.9），对 SQL 注入攻击输入位置的规律进行模型化表达，为 SQL 注入用例位置信息提供指导，同时 SGM 对此的描述方式又可避免攻击树建模的重复冗余和不一致问题。而目前相关研究中所提出的 SQL 注入攻击树模型，要么未描述 Web 应用 SQL 注入攻击输入位置的规律[208]；要么对攻击位置的描述过于具体，只能针对具体某个 Web 应用[209]，普遍指导意义不强。

（6）SGM 建模方法支持对子目的反作用（counteracting）的描述，能以此描述 Web 应用对 SQL 注入攻击的防御措施，为相关研究提供指

导信息，为其后的研究如基于模型的 SQL 注入安全漏洞防御等提供可扩展的接口。而攻击树建模方式不支持对攻击行为反作用的描述。

以上总结的本节所提出的基于 SGM 的 SQL 注入攻击模型优点中，除相比于攻击树建模方式在图形表达、方便理解方面的优点（如第一、二点）之外，更为重要的是其在表述 SQL 注入攻击位置、攻击输入、漏洞反应规律信息方面的优势（第三、四、五点），从而使本书所提出的 SQL 注入模型可以比目前相关研究中已经提出的 SQL 注入攻击模型[208,209]更全面地描述 SQL 注入攻击行为内在的规律信息，这使得所提出的新的 SQL 注入攻击模型比目前已有的模型更适宜满足前面所提出的测试框架对攻击模型指导优化 SQL 注入用例的要求，作为指导生成优化 SQL 用例的基础。因此，通过上述分析比较，采用本节提出的基于 SGM 的 SQL 注入攻击模型作为本章所提出的渗透测试框架的驱动模型是较为适合的。下面将以图 7.5～图 7.9 中的 SQL 注入模型作为指导模型，进行 SQL 注入用例建模和优化用例生成的研究。

7.3 SQL 注入渗透测试用例的形式化建模

本节根据 7.1 节所提出的渗透测试框架思想，在 7.2 节所提出的基于安全目的模型的 SQL 注入攻击模型指导下，对 SQL 注入渗透测试用例进行形式化建模，并对用例模型的实例化进行研究。在对 SQL 渗透

测试用例形式化建模的基础上，提出对所建用例模型的实例化方法，包括提出一系列渗透测试用例的覆盖准则等，将所建立的形式化模型转化为实际的 SQL 注入渗透测试用例，最终生成优化 SQL 注入用例集合。通过本章的研究工作，最终实现 7.2 节所提出的渗透测试框架中以形式化模型指导生成优化 SQL 注入用例的过程，体现本书中所建立的形式化模型方法对实际渗透测试的支持作用。

7.3.1　SQL 注入攻击输入建模

对于优化 SQL 注入用例而言，提高渗透测试准确度包括两项最为重要的研究内容：全面考虑可能的攻击输入、准确判定漏洞的存在性。这两项分别对应 SQL 注入用例输入和期望输出，本节中首先对前者进行研究。

7.2 节中使用攻击树和 SGM 对 SQL 注入安全漏洞进行了逻辑目的描述，这些模型的目的在于实现全面考虑攻击手段的研究目标：其实现了对于 SQL 注入攻击逻辑目的规律的描述，比非模型化的攻击手段描述或枚举攻击手段的方法，更有助于在渗透测试的模拟攻击中全面了解与考察 SQL 注入攻击安全漏洞。

在图 7.5～图 7.8 攻击模型的指导下，本节专门研究对不同种类的 SQL 注入攻击输入（SQL 注入用例输入）进行形式化建模问题，实现对 SQL 注入攻击规律的形式化描述。通过形式化符号和符号表达式构成 SQL 注入攻击输入模型，以模型表述 SQL 注入攻击输入的规律性，据此明确渗透测试中应使用什么用例，并能以有限规模方式描述这些用例。为此本节中提出定义注入攻击输入算子和算子表达式的方法，建立

对 SQL 注入用例输入集合的形式化描述体系。

首先定义注入攻击输入算子：▽。∇_i 代表某类攻击输入集合，如 ∇_{SQL} 代表 SQL 注入攻击输入总体集合。结合图 7.5～图 7.8 模型描述和前述 SQL 注入各类攻击手段，表 7.4 给出了各 SQL 注入攻击输入算子的定义。

表 7.4　注入攻击输入算子定义

算子	定义	示例*
∇_{DS}	异常字符集合	‘/“/;and user>0/admin' --/@@vesion;--
∇_{LG}	选取干扰认证机制的异常字符集合	admin' --/ ‘--/ /**/
∇_{DC}	异常命令集合	having 1=1--； / union select @@version；
∇_{CON}	条件式集合	=/<>/</>/<=/>=/IN/BETWEEN/LIKE/!=/ 1=(select count(*) from admin where len(name)> 5)；
∇_{IE}	选取恒等式（重言式）集合	1=1 / ‘d’<‘s’ / ‘b’ in (‘a’,‘b’) /　something’ like ‘some%’
∇_{NE}	选取不等式（矛盾式）集合	1=2/‘d’>‘s’/‘b’ not in (‘a’,‘b’)
∇_{TI}	利用条件式构造时间推断命令集合	if (*conditional*) waitfor delay '0:0:10' / SELECT pg_sleep(10) / IF (*conditional*) BENCHMARK(100,MD5(1))
∇_{COMS}	可执行命令集合	;alter …… add test date;-- /' ;exec master..xp_cmdshell ……
∇_{SQLC}	选取可执行 SQL 命令	; select * from …… / UNION select …… from ……; / ;insert into ……vaules(……)--
∇_{SP}	选取可执行存储过程命令	;exec master..xp_blank>_cmdshell dir / ;exec master..xp_cmdshell
∇_{AND}	以逻辑与关系注入条件式	AND 1=2
∇_{OR}	以逻辑或关系注入条件式	OR 49>45
∇_{disg}	对注入进行伪装	sEleCt / %73%45%6C%65%43%74

* 示例间以 “/ ” 分隔

根据上述注入参数算子，首先实现对 SQL 注入攻击输入集合的分类。在 Web 安全测试中，可据此将 SQL 注入输入划分为几类子元素，比随机枚举方式更有条理地表述 SQL 注入攻击。同时上述算子中也定义了相应类型攻击输入集合的操作，如∇_{DS}，∇_{DC}，∇_{CON}，∇_{COMS}是基本输入类型集合，∇_{LG}，∇_{IE}，∇_{NE}，∇_{TI}，∇_{SQLC}，∇_{SP}，∇_{AND}，∇_{OR}，∇_{disg}是对基本集合的操作算子。

进一步将算子结合起来形成算子表达式，以表述实现某种 SQL 注入攻击目的所需的攻击输入集合及其使用规律。为此定义算子间的操作符如下。

定义||为算子的或操作，&&为算子的与操作。$\nabla_1||\nabla_2=\{x|x\in\nabla_1\vee x\in\nabla_2\}$，对应 SGM 中的 OR 操作符，表示对于实现某攻击目的，∇_1和∇_2所代表的两种攻击输入可选其一（或都选）；$\nabla_1\&\&\nabla_2=\{x,y|x\in\nabla_1\wedge y\in\nabla_2\vee x\in\nabla_2\wedge y\in\nabla_1\}$，对应 SGM 的 AND 操作符，表示$\nabla_1$和$\nabla_2$所代表的两种攻击输入应同时使用。且$\nabla_1||\nabla_1=\nabla_1$。

定义算子的复合运算为 • ，算子∇_1与算子∇_2进行复合运算记为$\nabla_1\bullet\nabla_2$，表示∇_1处理算子∇_2代表的攻击输入集合，生成新的或复合的参数形式：$\{x|x\in\nabla_2\wedge\nabla_1\bullet x\}$。

上述算子之间运算符的优先级定义：复合运算符•的运算顺序为从右至左，即$\nabla_1\bullet\nabla_2\bullet\nabla_3$意为$\nabla_1\bullet(\nabla_2\bullet\nabla_3)$。&&操作符优先级高于||操作符，即$\nabla_1||\nabla_2\ \&\&\nabla_3$等价于$\nabla_1||(\nabla_2\&\&\nabla_3)$。复合运算符•的优先级高于&&和||操作符，括号的优先级最高。基于上述算子定义和图 7.5～图 7.8 的 SGM 描述，将 SQL 注入攻击分为 5 种，分别描述其攻击输入规律，如表 7.5 所示。

表 7.5　注入攻击输入算子表达式

SQL 注入攻击目的	攻击输入算子表达式
错误信息利用	$\nabla_{DS}\|\|\nabla_{DC}\|\|\nabla_{disg}\bullet(\nabla_{DS}\|\|\nabla_{DC})$
盲注入	$\nabla_{TI}\bullet\nabla_{CON}\|\|\nabla_{AND}\bullet\nabla_{IE}\bullet\nabla_{CON}\&\&\nabla_{AND}\bullet\nabla_{NE}\bullet\nabla_{CON}\|\|\nabla_{disg}\bullet(\nabla_{TI}\bullet\nabla_{CON}\|\|\nabla_{AND}\bullet\nabla_{IE}\bullet\nabla_{CON}\&\&\nabla_{AND}\bullet\nabla_{NE}\bullet\nabla_{CON})$
注入运行 SQL 命令	$\nabla_{SQLC}\bullet\nabla_{COMS}\|\|\nabla_{OR}\bullet\nabla_{IE}\bullet\nabla_{CON}\|\|\nabla_{disg}\bullet(\nabla_{SQLC}\bullet\nabla_{COMS}\|\|\nabla_{OR}\bullet\nabla_{IE}\bullet\nabla_{CON})$
注入运行存储过程	$\nabla_{SP}\bullet\nabla_{COMS}\|\|\nabla_{disg}\bullet(\nabla_{SP}\bullet\nabla_{COMS})$
绕过认证	$\nabla_{OR}\bullet\nabla_{IE}\bullet\nabla_{CON}\|\|\nabla_{LG}\bullet\nabla_{DS}\|\|\nabla_{disg}\bullet(\nabla_{OR}\bullet\nabla_{IE}\bullet\nabla_{CON}\|\|\nabla_{LG}\bullet\nabla_{DS})$

上述算子规则式的定义中，既反映了 SQL 注入攻击输入使用规律，又反映了攻击输入之间的分类归属。所给出的定义中，异常字符集合（∇_{DS}）、异常命令集合（∇_{DC}）、条件式集合（∇_{CON}）、可执行命令集合（∇_{COMS}）是 SQL 注入攻击输入全集的 4 个基本大类，这一点如图 7.5 模型最上端对攻击输入的描述所示。表 7.5 中盲注入攻击算子规则式表述注入恒等式（$\nabla_{IE}\bullet\nabla_{CON}$）与不等式（$\nabla_{NE}\bullet\nabla_{CON}$）或时间推断命令（$\nabla_{TI}\bullet\nabla_{CON}$）、注入运行 SQL 命令和绕过认证攻击的规则式描述都需要注入恒等式（$\nabla_{IE}\bullet\nabla_{CON}$），从其规则式的表达方式上可知这几种不同的攻击目的所使用的攻击输入之间的分类联系，其输入都属于条件式（∇_{CON}）的攻击输入大类。同样在表 7.5 中注入运行 SQL 命令和注入运行存储过程攻击子目的中用到可执行命令集合（∇_{COMS}）的子类：可执行 SQL 命令（$\nabla_{SQLC}\bullet\nabla_{COMS}$）与存储过程命令（$\nabla_{SP}\bullet\nabla_{COMS}$），表明两者都属于可执行命令这一大类。错误信息利用注入异常字符∇_{DS}和绕过认证注入用于干扰登录认证机制的异常字符（$\nabla_{LG}\bullet\nabla_{DS}$）都属于异常字符集合（$\nabla_{DS}$）这一大类。

以上所定义的参数算子可以较好地形式化表述 SQL 注入攻击参数

使用规则。但从直观形态来看上述规则式的符号重复率较高，形式不够简洁。为此进一步定义关于算子表达式的运算规则，目的是在其表达内容意义不变的情况下，将原算子表达式化简为较为简洁的形式。

根据上述攻击参数算子的含义和操作符的定义，定义算子表达式的运算规则，如表 7.6 所示。

表 7.6　算子表达式运算规则定义

定律	定义（是否一定成立）		
交换律	$\nabla_1 \| \| \nabla_2 = \nabla_2 \| \| \nabla_1$	$\nabla_1 \&\& \nabla_2 = \nabla_2 \&\& \nabla_1$	$\nabla_1 \bullet \nabla_2 \neq \nabla_2 \bullet \nabla_1$
结合律	$(\nabla_1 \| \| \nabla_2) \| \| \nabla_3 = \nabla_1 \| \| (\nabla_2 \| \| \nabla_3)$	$(\nabla_1 \&\& \nabla_2) \&\& \nabla_3 = \nabla_1 \&\& (\nabla_2 \&\& \nabla_3)$	$(\nabla_1 \bullet \nabla_2) \bullet \nabla_3 \neq \nabla_1 \bullet (\nabla_2 \bullet \nabla_3)$
分配律	$\nabla_1 \bullet (\nabla_2 \| \| \nabla_3) = \nabla_1 \bullet \nabla_2 \| \| \nabla_1 \bullet \nabla_3$	$(\nabla_1 \| \| \nabla_2) \bullet \nabla_3 = \nabla_1 \bullet \nabla_3 \| \| \nabla_2 \bullet \nabla_3$	$(\nabla_1 \&\& \nabla_2) \bullet \nabla_3 = \nabla_1 \bullet \nabla_3 \&\& \nabla_2 \bullet \nabla_3$

交换律证明：

$\because \nabla_1 \| \| \nabla_2 = \{x \mid x \in \nabla_1 \vee x \in \nabla_2\}$

$\nabla_2 \| \| \nabla_1 = \{x \mid x \in \nabla_2 \vee x \in \nabla_1\}$

$\therefore \nabla_1 \| \| \nabla_2 = \nabla_2 \| \| \nabla_1$

$\because \nabla_1 \&\& \nabla_2 = \{x,y \mid x \in \nabla_1 \wedge y \in \nabla_2 \vee x \in \nabla_2 \wedge y \in \nabla_1\}$

$\nabla_2 \&\& \nabla_1 = \{x,y \mid x \in \nabla_2 \wedge y \in \nabla_1 \vee x \in \nabla_1 \wedge y \in \nabla_2\}$

$\therefore \nabla_1 \&\& \nabla_2 = \nabla_2 \&\& \nabla_1$

$\because \nabla_1 \bullet \nabla_2$ 表示 ∇_1 处理算子 ∇_2 代表的注入参数，生成新的或复合的参数形式

$\therefore \nabla_1$ 是处理操作算子，∇_2 是被处理输入集合，$\nabla_2 \bullet \nabla_1$ 不一定有意义

$\therefore \nabla_1 \bullet \nabla_2 \neq \nabla_2 \bullet \nabla_1$

结合律证明：

$\because (\nabla_1 \| \| \nabla_2) \| \| \nabla_3 = \{x \mid x \in \nabla_1 \vee x \in \nabla_2 \vee x \in \nabla_3\}$

$\nabla_1||(\nabla_2||\nabla_3)=\{x|x\in\nabla_1\vee x\in\nabla_2\vee x\in\nabla_3\}$

$\therefore(\nabla_1||\nabla_2)||\nabla_3=\nabla_1||(\nabla_2||\nabla_3)$

$\because(\nabla_1\&\&\nabla_2)\&\&\nabla_3=\{x,y,z|x\in\nabla_1\wedge y\in\nabla_2\wedge z\in\nabla_3\vee x\in\nabla_1\wedge y\in\nabla_3\wedge z\in\nabla_2\vee x\in\nabla_2\wedge y\in\nabla_1\wedge z\in\nabla_3\vee x\in\nabla_2\wedge y\in\nabla_3\wedge z\in\nabla_1\vee x\in\nabla_3\wedge y\in\nabla_1\wedge z\in\nabla_2\vee x\in\nabla_3\wedge y\in\nabla_2\wedge z\in\nabla_1\}$

$\nabla_1\&\&(\nabla_2\&\&\nabla_3)=\{x,y,z|x\in\nabla_1\wedge y\in\nabla_2\wedge z\in\nabla_3\vee x\in\nabla_1\wedge y\in\nabla_3\wedge z\in\nabla_2\vee x\in\nabla_2\wedge y\in\nabla_1\wedge z\in\nabla_3\vee x\in\nabla_2\wedge y\in\nabla_3\wedge z\in\nabla_1\vee x\in\nabla_3\wedge y\in\nabla_1\wedge z\in\nabla_2\vee x\in\nabla_3\wedge y\in\nabla_2\wedge z\in\nabla_1\}$

$\therefore(\nabla_1\&\&\nabla_2)\&\&\nabla_3=\nabla_1\&\&(\nabla_2\&\&\nabla_3)$

$\because\nabla_1\bullet\nabla_2\neq\nabla_2\bullet\nabla_1$

$\therefore(\nabla_1\bullet\nabla_2)\bullet\nabla_3$ 有意义而 $\nabla_1\bullet(\nabla_2\bullet\nabla_3)$ 不一定有意义，反之亦然

$\therefore(\nabla_1\bullet\nabla_2)\bullet\nabla_3\neq\nabla_1\bullet(\nabla_2\bullet\nabla_3)$

分配律证明：

$\because\nabla_2||\nabla_3=\{x|x\in\nabla_2\vee x\in\nabla_3\}$

$\nabla_1\bullet(\nabla_2||\nabla_3)=\{\nabla_1\bullet x|x\in\nabla_2\vee x\in\nabla_3\}$

$\nabla_1\bullet\nabla_2||\nabla_1\bullet\nabla_3=\{\nabla_1\bullet x|x\in\nabla_2\vee x\in\nabla_3\}$

$\therefore\nabla_1\bullet(\nabla_2||\nabla_3)=\nabla_1\bullet\nabla_2||\nabla_1\bullet\nabla_3$

$\because(\nabla_1||\nabla_2)\bullet\nabla_3=\{x|x\in\nabla_3\wedge\nabla_1\bullet x\vee x\in\nabla_3\wedge\nabla_2\bullet x\}$

$\nabla_1\bullet\nabla_3||\nabla_2\bullet\nabla_3=\{x|x\in\nabla_3\wedge\nabla_1\bullet x\vee x\in\nabla_3\wedge\nabla_2\bullet x\}$

$\therefore(\nabla_1||\nabla_2)\bullet\nabla_3=\nabla_1\bullet\nabla_3||\nabla_2\bullet\nabla_3$

$\because(\nabla_1\&\&\nabla_2)\bullet\nabla_3=\{x,y|x\in\nabla_3\wedge y\in\nabla_3\wedge\nabla_1\bullet x\wedge\nabla_2\bullet y\vee x\in\nabla_3\wedge y\in\nabla_3\wedge\nabla_2\bullet x\wedge\nabla_1\bullet y\}$

$\nabla_1\bullet\nabla_3\ \&\&\nabla_2\bullet\nabla_3=\{x,y|x\in\nabla_3\wedge\nabla_1\bullet x\wedge y\in\nabla_3\wedge\nabla_2\bullet y\vee x\in\nabla_3\wedge\nabla_2\bullet x\wedge y\in\nabla_3\wedge\nabla_1\bullet y\}$

$\therefore (\nabla_1 \&\& \nabla_2)\bullet\nabla_3 = \nabla_1\bullet\nabla_3 \&\& \nabla_2\bullet\nabla_3$

定义Φ为空算子符号，表示对算子▽不进行任何操作，本身不含任何元素。其作用类似于普通代数运算式中的“1”，用来指代算子▽自身。例如：$\nabla_1 || \Phi = \nabla_1 \&\& \Phi = \Phi\bullet\nabla_1 = \nabla_1\bullet\Phi = \nabla_1$。在规则式中Φ可理解为“不进行任何操作”。根据上述定义，可将原列举出的攻击参数规则式化简为更为简洁的形式。

例如，表 7.5 中第三行的盲注入攻击 blindInject(WA)的试探参数规则表达式，依据上述定义和规则，其化简过程如下：

$\nabla_{TI}\bullet\nabla_{CON} || \nabla_{AND}\bullet\nabla_{IE}\bullet\nabla_{CON} \&\& \nabla_{AND}\bullet\nabla_{NE}\bullet\nabla_{CON} || \nabla_{disg}\bullet(\nabla_{TI}\bullet\nabla_{CON} || \nabla_{AND}\bullet\nabla_{IE}\bullet\nabla_{CON} \&\& \nabla_{AND}\bullet\nabla_{NE}\bullet\nabla_{CON})$

$= \Phi\bullet(\nabla_{TI}\bullet\nabla_{CON} || (\nabla_{AND}\bullet\nabla_{IE}\bullet\nabla_{CON} \&\& \nabla_{AND}\bullet\nabla_{NE}\bullet\nabla_{CON})) || \nabla_{disg}\bullet(\nabla_{TI}\bullet\nabla_{CON} || (\nabla_{AND}\bullet\nabla_{IE}\bullet\nabla_{CON} \&\& \nabla_{AND}\bullet\nabla_{NE}\bullet\nabla_{CON}))$

$= (\Phi || \nabla_{disg})\bullet(\nabla_{TI}\bullet\nabla_{CON} || (\nabla_{AND}\bullet\nabla_{IE}\bullet\nabla_{CON} \&\& \nabla_{AND}\bullet\nabla_{NE}\bullet\nabla_{CON}))$

$= (\Phi || \nabla_{disg})\bullet(\nabla_{TI}\bullet\nabla_{CON} || (\nabla_{AND}\bullet\nabla_{IE} \&\& \nabla_{AND}\bullet\nabla_{NE})\bullet\nabla_{CON})$

$= (\Phi || \nabla_{disg})\bullet((\nabla_{TI} || (\nabla_{AND}\bullet\nabla_{IE} \&\& \nabla_{AND}\bullet\nabla_{NE}))\bullet\nabla_{CON})$

化简后规则式表达的意义不变，在形式上更加简洁。根据上述相关符号定义，用自然语言表达，此规则式的含义为：选取条件式集合中的时间推断命令，或者选取其中的恒等式与不等式一起使用（进行逻辑与注入），必要时可采用这些命令的伪装形式，注入测试 Web 应用是否可进行盲注入攻击。

对表 7.5 中的 SQL 注入攻击参数规则表达式进行化简后的形式如表 7.7 所示。对比两表可看出，通过本节的规则表达式化简规则定义，对完整的初始表达式进行了有效的化简，以简洁形式表述。同时，化简后的规则表达式表述意义不变，不丢失信息，而表现形式更为简洁。这

种思路为渗透测试实际应用算子表达式提供了良好的条件。更为重要的是，所提出的运算规则可使算子表达式进行代数式一般的运算，从而实现算子表达式合并运算，将一系列表达式转化为一个最终式，进行核对用例完整性等方面的操作。

表 7.7　化简后的算子表达式

SQL 注入攻击目的	化简后的攻击输入算子表达式
错误信息利用	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{DS} \| \| \triangledown_{DC})$
盲注入	$(\Phi \| \| \triangledown_{disg}) \bullet ((\triangledown_{TI} \| \| (\triangledown_{AND} \bullet \triangledown_{IE} \&\& \triangledown_{AND} \bullet \triangledown_{NE})) \bullet \triangledown_{CON})$
注入运行 SQL 命令	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{SQLC} \bullet \triangledown_{COMS} \| \| \triangledown_{OR} \bullet \triangledown_{IE} \bullet \triangledown_{CON})$
注入运行存储过程	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{SP} \bullet \triangledown_{COMS})$
绕过认证	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{OR} \bullet \triangledown_{IE} \bullet \triangledown_{CON} \| \| \triangledown_{LG} \bullet \triangledown_{DS})$

以下应用对算子规则式上述运算规则的定义，验算在一定的 SQL 注入攻击输入分类表述体系下，所列举的攻击输入是否全面地覆盖到已知的攻击输入形式。下面对此进行证明。

定理 7.1　在图 7.5～图 7.8 的 SGM 表述下，表 7.7 中的注入攻击输入算子表达式集合是对 SQL 注入攻击输入的全面描述。

证明：表 7.7 中 SQL 注入 5 种子目的攻击手段（错误信息利用、盲注入、绕过认证、注入运行 SQL 命令、注入运行存储过程）所用到的攻击输入并集为

$$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{DS} \| \| \triangledown_{DC}) \| (\Phi \| \| \triangledown_{disg}) \bullet ((\triangledown_{TI} \| \| (\triangledown_{AND} \bullet \triangledown_{IE} \&\& \triangledown_{AND} \bullet \triangledown_{NE})) \bullet \triangledown_{CON}) \| (\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{OR} \bullet \triangledown_{IE} \bullet \triangledown_{CON} \| \| \triangledown_{LG} \bullet \triangledown_{DS}) \| (\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{SQLC} \bullet \triangledown_{COMS} \| \| \triangledown_{OR} \bullet \triangledown_{IE} \bullet \triangledown_{CON}) \| (\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{SP} \bullet \triangledown_{COMS}) \tag{7.1}$$

消除用于实际测试的攻击输入伪装算子 $\triangledown_{disg}$ 及其复合操作，忽略

∇_{AND}和∇_{OR}操作符及其复合操作，只考察攻击输入的并集，式（7.1）可化为

$$\nabla_{DS}||\nabla_{DC}||((\nabla_{TI}||(\nabla_{IE}\&\&\nabla_{NE}))\bullet\nabla_{CON})||\nabla_{IE}\bullet\nabla_{CON}||\nabla_{LG}\bullet\nabla_{DS}$$
$$||\nabla_{SQLC}\bullet\nabla_{COMS}||\nabla_{IE}\bullet\nabla_{CON}||\nabla_{SP}\bullet\nabla_{COMS} \quad (7.2)$$

根据表 7.4 注入攻击输入算子的定义和表 7.6 算子表达式运算规则定义，式（7.2）可化为

$$\nabla_{DS}||\nabla_{LG}\bullet\nabla_{DS}||\nabla_{DC}||\nabla_{IE}\bullet\nabla_{CON}||$$
$$(\nabla_{SQLC}||\nabla_{SP})\bullet\nabla_{COMS}||(\nabla_{TI}||(\nabla_{IE}\&\&\nabla_{NE}))\bullet\nabla_{CON} \quad (7.3)$$

根据图 7.5、图 7.8 的 SGM 描述，可定义

$$\nabla_{LG}\bullet\nabla_{DS}\subseteq\nabla_{DS}$$

$\nabla_{DS}||\nabla_{LG}\bullet\nabla_{DS}=\nabla_{DS}$

式（7.3）可变为

$$\nabla_{DS}||\nabla_{DC}||(\nabla_{SQLC}||\nabla_{SP})\bullet\nabla_{COMS}||(\nabla_{IE}||\nabla_{TI}||(\nabla_{IE}\&\&\nabla_{NE}))\bullet\nabla_{CON} \quad (7.4)$$

操作符||和&&的定义包含∪功能，在攻击输入并集生成时可以∪代替二者。根据图 7.5～图 7.8 的 SGM 描述，已知：

$\vdash \nabla_{CON}=\nabla_{IE}\bullet\nabla_{CON}\cup\nabla_{NE}\bullet\nabla_{CON}\cup\nabla_{TI}\bullet\nabla_{CON}$

$\vdash \nabla_{COMS}=\nabla_{SQLC}\bullet\nabla_{COMS}\cup\nabla_{SP}\bullet\nabla_{COMS}$

$\vdash \nabla_{SQL}=\nabla_{DS}\cup\nabla_{DC}\cup\nabla_{CON}\cup\nabla_{COMS}$

式（7.4）可化为

$$\nabla_{DS}||\nabla_{DC}||\nabla_{COMS}||\nabla_{CON} \quad (7.5)$$

式（7.5）可化为

$$\nabla_{DS}\cup\nabla_{DC}\cup\nabla_{COMS}\cup\nabla_{CON}=\nabla_{SQL} \quad (7.6)$$

命题得证。

在一个 SQL 注入攻击输入分类体系前提下，如果对 SQL 注入攻击输入的描述其并集不为∇_{SQL}，即不能构成对 SQL 注入攻击输入的全面

描述，就说明其不符合对前述优化的渗透测试用例全面性要求。这说明本研究所提出的算子形式化 SQL 注入攻击输入模型具有验算 SQL 注入用例是否全面的功能。

对于目前相关研究中[200,206,210]分类体系下的各种 SQL 注入攻击方式，本节提出的注入攻击输入算子也可以对其攻击输入进行描述。

① 重言式攻击：$\nabla_{OR}\bullet\nabla_{IE}\bullet\nabla_{CON}$

② 联合查询、多命令语句攻击：$\nabla_{SQLC}\bullet\nabla_{COMS}$

③ 异常命令语句攻击：∇_{DC}

④ 推断攻击：$(\nabla_{TI}\|(\nabla_{AND}\bullet\nabla_{IE}\&\&\nabla_{AND}\bullet\nabla_{NE}))\bullet\nabla_{CON}$

⑤ 攻击输入编码伪装：$\nabla_{disg}\bullet\cdots$

⑥ 存储过程攻击：$\nabla_{SP}\bullet\nabla_{COMS}$

从表 7.7 中可见，上述攻击输入形式已经包括在我们所总结的注入攻击输入算子规则式中，即表 7.7 中所提出的规则式是对当前相关研究[206,210]中所总结的 SQL 注入攻击输入的全面描述。相比相关研究中对攻击输入的非形式化自然语言描述，本节提出的算子规则式以简洁的少数符号即实现了对其的描述，并且表达了具体的攻击输入使用规则等信息。

在文献[199]中提出了以形式化的算子符号表达漏洞测试用例的方法，但对于 SQL 注入安全漏洞测试，其所定义的算子并未描述具体的 SQL 注入攻击种类，只以一个算子符号代表所有的 SQL 注入攻击输入，表述粒度过粗，而且未如本研究所提方法基于攻击模型指导生成对 SQL 用例输入分类的表述，因此对 SQL 注入渗透测试指导性不足。基于本节提出的 SQL 注入攻击输入算子及其表达式，进一步建立 SQL 注入用例模型，从攻击输入、预期输出和攻击位置等方面模型化描述其渗透测试用例。

7.3.2　SQL 注入安全漏洞特征的形式化描述

在对 SQL 注入攻击输入形式化建模的基础上，本节对另一项通过优化用例提高渗透测试准确度重要的研究内容——准确判定漏洞的存在性问题进行研究。这实际上是建立 SQL 注入用例期望输出规则问题。

本节采用形式化语言对 SQL 注入漏洞特征建模。用形式化语言定义 SQL 注入攻击漏洞的行为特征，即以形式化语言描述 SQL 注入各种攻击输入规律，以及 Web 应用对各种攻击输入有怎样的反应特征则可判定其具有 SQL 注入安全漏洞。这项研究的目的和意义在于，为 SQL 注入渗透测试准确判定 Web 应用 SQL 注入漏洞存在性(用例输出建模)提供指导。

给出相关的形式化符号定义。设攻击者为 attacker，Web 应用记为 WA，attacker.input 表示攻击者向 WA 提交的攻击输入，attacker.GET_knowledge()表示攻击者可获得有利于攻击的信息，WA.response()表示 WA 对输入的反应，WA.response().error 表示 WA 产生出错信息，WA.response().run 表示 Web 应用对攻击者注入的命令是否予以执行，WA.response().state 表示 WA 的反应状态，authenticated 表示是否通过 Web 应用登录认证，usr,pwd 表示输入 WA 的用户名和密码信息。

根据上一节对于 SQL 注入攻击输入算子的设定和图 7.5～图 7.8 攻击模型的描述，做出如下定义。

定义 7.1　SQL 注入的错误信息利用，记为 deformSInject(WA)。表示攻击者可通过注入异常字符或无法执行的命令来诱发 Web 应用产生错误信息，从中获得有价值的信息，

$$\text{deformSInject(WA)} \leftrightarrow (\text{attacker.input} \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{DS} \| \bigtriangledown_{DC})) \wedge \text{information} \in \text{WA.response (attacker.input).error} \wedge \text{attacker. GET\_knowledge (information);} \quad (7.7)$$

定义 7.2 SQL 注入盲注入，记为 blindInject(WA)。如图 7.6 所示，其包括时间推断 timing_inference()和条件猜解 condition_inference()两种方式：

$$\text{blindInject(WA)} \leftrightarrow \text{timing\_inference(WA)} \vee \text{condition\_inference(WA);} \quad (7.8)$$

时间推断攻击表示攻击者可构造包含条件式的时间延迟命令作为攻击输入，如 Web 应用对所注入的命令予以执行，表示存在此种安全漏洞：

$$\text{timing\_inference(WA)} \leftrightarrow (\text{attacker.input} \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{TI} \bullet \bigtriangledown_{CON})) \wedge \text{WA.response (attacker. input). run==true;} \quad (7.9)$$

条件猜解攻击表示攻击者构造一系列成立及不成立的条件式并将其以逻辑“与”的关系注入 Web 应用，观察 Web 应用对条件式成立与否的不同反应，以推断窃取系统信息：

$$\text{condition\_inference(WA)} \leftrightarrow ((\text{attacker.input}i \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{AND} \bullet \bigtriangledown_{IE} \bullet \bigtriangledown_{CON}) \wedge \text{attacker.input } j \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{AND} \bullet \bigtriangledown_{NE} \bullet \bigtriangledown_{CON})) \vee (\text{attacker.input } i \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{AND} \bullet \bigtriangledown_{NE} \bullet \bigtriangledown_{CON}) \wedge \text{attacker.input } j \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{AND} \bullet \bigtriangledown_{IE} \bullet \bigtriangledown_{CON}))) \wedge \text{WA.response(attacker.input i).state} \neq \text{WA. response (attacker.input j).state;} \quad (7.10)$$

定义 7.3 SQL 注入窃取系统信息，记为 I(WA)，其包括错误信息利用和盲注入两种方式：

$$\text{I(WA)} \leftrightarrow \text{deformSInject(WA)} \vee \text{blindInject(WA)} \quad (7.11)$$

定义 7.4 SQL 注入运行恶意命令攻击，表示攻击者可向 WA 注入 SQL 语法允许的、符合攻击者目的的命令并运行，记为 R(WA)，其包括注入 SQL 命令攻击和注入运行存储过程攻击：

$$\text{R(WA)} \leftrightarrow \text{SQLRuning(WA)} \vee \text{SPRuning(WA);} \quad (7.12)$$

其中，SQLRuning(WA)表示 Web 应用注入 SQL 命令攻击，SPRuning(WA)表示注入运行存储过程攻击，即

$$\text{SQLRuning(WA)} \leftrightarrow (\text{attacker.input} \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{SQLC} \bullet \bigtriangledown_{COMS} \| \bigtriangledown_{OR} \bullet \bigtriangledown_{IE} \bullet \bigtriangledown_{CON})) \wedge \text{WA.response(attacker.input).run==true;} \quad (7.13)$$

$$\text{SPRuning(WA)} \leftrightarrow \text{attacker.input} \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{SP} \bullet \bigtriangledown_{COMS}) \wedge \text{WA.response(attacker.input). run==true;} \quad (7.14)$$

定义 7.5　SQL 注入绕过认证，表示攻击者可以通过 SQL 注入攻击实现绕过 Web 应用 WA 的合法用户认证，记为 L(WA)。

$$\text{L(WA)} \leftrightarrow (\text{attacker.input} \in (\Phi \| \bigtriangledown_{disg}) \bullet (\bigtriangledown_{OR} \bullet \bigtriangledown_{IE} \bullet \bigtriangledown_{CON} \| \bigtriangledown_{LG} \bullet \bigtriangledown_{DS})) \wedge \text{usr,pwd} \in \text{attacker.input} \wedge \text{WA.response(usr, pwd).authenticated==true} \quad (7.15)$$

定义 7.6　某 WA 存在 SQL 注入安全漏洞，记为 SQLI(WA)，则

$$\text{SQLI(WA)} \leftrightarrow \text{I(WA)} \vee \text{R(WA)} \vee \text{L(WA)} \quad (7.16)$$

本部分对 SQL 注入的漏洞特征进行了形式化定义。所建立的 SQL 注入安全漏洞特征表达式是从 Web 应用外部特征视角即黑盒测试方式出发对 SQL 注入漏洞的特征进行描述，在文献[223]中也给出了关于 Web 应用中命令注入类攻击的形式化定义，但其形式化定义是从 Web 应用代码内部特征（如攻击输入所生成的 SQL 命令语句 parse tree）的角度对注入攻击进行定义，且其给出的定义所面向的研究问题是如何识别与防御注入攻击。与其相比，本研究所提出的上述 SQL 注入特征形式化式无须关注 Web 应用内部代码执行情况及其特征分析，只需要从 Web 应用对外反应判定 SQL 注入安全漏洞存在性，因此更加适用于以黑盒方式为特征的渗透测试研究问题。

通过上述形式化描述，一方面对当前 SQL 注入攻击输入集合进行了形式化描述和分类，克服了目前相关研究中对其随机枚举描述方式的无规律性和无限性；另一方面定义了 SQL 注入安全漏洞特征的形式化

描述式，为判定 Web 应用的 SQL 注入漏洞存在性提供了必要的准则。且式（7.7）～式（7.16）采用的形式化描述方法为类编程的表达方式，这有助于实际渗透测试系统编程实现时，直接根据这种表达方式编写判定 Web 应用 SQL 注入漏洞存在性反应的函数子程序，从而将形式化理论与测试实践结合。

7.3.3 SQL 注入用例建模

本节在上述研究的基础上实现对 SQL 注入渗透测试用例建模。在本书所提出的渗透测试框架中，攻击模型对下端渗透测试的支持体现在三个方面：提供 SQL 注入攻击位置、攻击输入和漏洞存在性判定准则的信息，因此本节在以上工作形式化建模的基础上，建立包含有 SQL 注入攻击位置、攻击输入和漏洞反应规律的渗透测试用例模型。

在对于软件测试用例的相关研究中，测试用例一般形式化定义为一个三元组[214]：t= (Pre,In,Out)，其中 Pre 为前置条件，In 为输入，Out 为期望输出。

对于 SQL 注入安全漏洞的渗透测试用例，在前面对 SQL 注入渗透测试用例定义的基础上，进一步定义 SQL 注入用例为上述三元组形式。其中 Pre 定义为 Web 应用输入点类型（找到这些输入点是攻击的前提），定义 In 为攻击输入集合，定义 Out 为软件存在 SQL 注入安全漏洞的反应特征。即定义 SQL 注入渗透测试用例至少包含三元组的这三部分信息。本研究的重点是 In 与 Out 的建模和实例化，所以对于 Pre 根据测试经验只考虑 Web 应用中的带参数 URL（形如/NewsList.aspx?id=12）与登录表单两类输入点作为测试对象。基于以上定义，给出对 SQL 注

入用例集合的形式化描述：用例集 TC=<AS_1, AS_2,……AS_N>，其中 AS_i 表示攻击模式，定义如下。

定义 7.7　攻击模式（attack scheme）为四元组 <OBJ,PRE,IN,OUT>，其中 OBJ 为攻击目的，PRE 为攻击输入点类型，IN 为攻击输入，OUT 为 Web 应用存在 SQL 注入安全漏洞的反应。

根据图 7.5～图 7.8 的模型和上述形式化描述式定义[式（7.7）～式（7.16）]，我们建立 SQL 注入渗透中测试用例模型，如表 7.8 所示。我们将 SQL 注入渗透测试用例集分为表 7.8 中的 5 种攻击模式（表中每行为一个 AS_i）。

表 7.8　SQL 注入渗透测试用例模型

OBJ	PRE	IN	OUT
错误信息利用	带参数 URL，登录表单	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{DS} \| \| \triangledown_{DC})$	deformSInject(WA)
盲注入（条件猜解）	带参数 URL	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{AND} \bullet \triangledown_{IE} \,\&\&\, \triangledown_{AND} \bullet \triangledown_{NE}) \bullet \triangledown_{CON})$	condition_inference(WA)
注入 SQL 命令	带参数 URL，登录表单	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{SQLC} \bullet \triangledown_{COMS} \| \| \triangledown_{OR} \bullet \triangledown_{IE} \bullet \triangledown_{CON})$	SQLRuning(WA)
注入运行存储过程	带参数 URL，登录表单	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{SP} \bullet \triangledown_{COMS})$	SPRuning(WA)
绕过认证	登录表单	$(\Phi \| \| \triangledown_{disg}) \bullet (\triangledown_{OR} \bullet \triangledown_{IE} \bullet \triangledown_{CON} \| \| \triangledown_{LG} \bullet \triangledown_{DS})$	L(WA)

所建立的渗透测试用例模型根据前述的 SQL 注入攻击输入建模和 SQL 注入安全漏洞特征形式化描述，对渗透测试中应使用的 SQL 注入用例进行形式化描述。模型指导渗透测试中，对 Web 应用依次进行各攻击模式（即 SQL 注入的子类型）的模拟攻击来确定 SQL 注入安全漏洞存在性。其中盲注入攻击模式（子类型）在此只考虑条件猜解攻击作为用例，因为时间推断攻击是要与条件式相结合才能实施的输入，所以

用例以条件猜解攻击确定条件式攻击输入的可输入性，即可确定时间推断攻击的可行性。这种以输入样式确定 SQL 注入用例等价性的思想，在以下用例模型实例化问题中会进行分析。

根据以上设定给出对 Web 应用进行 SQL 注入漏洞渗透测试算法。首先将用例集合中 IN 和 OUT 形式化描述根据一定的充分性准则和软件实际情况实例化为实际用例。对每种攻击模式，将其每个实例化的用例输入注入此目的 PRE 所述的输入点，判断 Web 的反应是否符合此目的 OUT 所述的特征（根据实例化 OUT），若符合则判定此输入点存在安全漏洞，记载此输入点位置及对其可注入的用例信息，最后将此信息集合作为测试结果返回。

定义 7.8 对于 Web 应用某输入点 p、攻击模式 AS_i，若将 AS_i.IN 模型的某实例化用例 inp 输入 p，使 Web 应用出现 AS_i.OUT 所描述的反应，则称 p 对 inp 是可注入的，记为 inp$\propto$p。

根据以上定义给出对 Web 应用进行 SQL 注入安全漏洞渗透测试算法。

算法 7.1 基于模型的 Web 应用 SQL 注入渗透测试算。

输入：受测 Web 应用，渗透测试用例集合 TC 形式化描述。

输出：受测 Web 系统中存在 SQL 注入安全漏洞的输入点及其可注入用例的信息集合 VIP。

① VIP=Φ；

② 爬行/分析受测 Web 系统，查找 TC 中 PRE 所述类型的输入点集合；

③ 根据 Web 应用实际情况和覆盖准则实例化 TC 中 IN 及 OUT 形式化描述，生成实例化用例集合 IN′和 OUT′；

④ Foreach 攻击模式 $AS_i \in TC$ ： // 每种攻击模式

⑤ {Foreach 输入点 $p_i \in AS_i$. PRE：

⑥　{Foreach 实例化用例 inp∈AS_i.IN′：

⑦　{将 inp 注入 p_i;

⑧　　if (inp∝p) // Web 应用的反应符合 AS_i.OUT′描述的特征

⑨　　{将 p_i 和 inp 的信息存入 VIP;} //endif

⑩　} // end of Foreach 实例化用例 inp

⑪　} // end of Foreach 输入点 p_i

⑫　} // end of 攻击模式 AS_i

⑬ **return** VIP

设测试用例集合 TC 中包含的 AS_i 数目为 m，Web 应用中所包含的 SQL 注入可攻击注入点总数目为 n，TC 中实例化用例集合 IN′中元素总数目为 q，AS_i 中 IN′的元素数量为 q_i，AS_i 中 PRE 的元素数量为 n_i，则算法 7.1 中步骤②时间复杂度约为 $\sum_{i=1}^{m} n_i \approx O(mn)$，步骤③时间复杂度约为 $O(q)$，步骤④～⑬时间复杂度约为 $\sum_{i=1}^{m} n_i q_i \approx O(mnq)$，故算法 7.1 时间复杂度约为 $O(mnq)$。实际测试中 mnq 都应是有限的集合元素数目，因此算法可在有限时间内完成。

算法 7.1 体现了上述提高渗透测试准确度的研究思想：向测试提供有序全面的用例输入（步骤③）和与其相匹配的明确的安全漏洞特征定义（步骤⑧）来判断 Web 应用的漏洞存在性，主要以这两方面的工作提高渗透测试准确度。

第 8 章

结　　语

由于网构软件自身的开放性与动态性及其运行环境的复杂性与多变性，可信性保证成为支持网构软件系统正常运行和有效服务的重要手段。本书通过对现有网构软件可信性保障与评估方法的分析，从网构软件的实体、实体间协同及软件系统三个层面提出了可信性保障的机制与方法，建立了网构软件实体间有保障的信任链，并根据网构软件开发与运行的特点提出了有针对性的可信性评估模型。

目前，网构软件可信性保障方法存在三方面的不足：信任信息主要来源于对网构软件实体进行交互的历史结果，推荐信息质量的度量依赖于事后评估的方式，信任传递参数对实体个体所处环境差异考虑不足。而传统的软件可信性评估方法也难以适应网构软件开放、动态的特点。针对上述问题，本书进行了如下研究。

（1）在网构软件实体层面，通过对现有网构软件实体模型的研究与分析，提出了具有自省性、自明性、自主性特点的网构软件强可信智能实体模型。通过功能与可信性保障分离的设计，保证了自省机制的灵活性及业务功能的安全性。定义了实体可信情况形式化描述语言EDSADL，为网构软件实体内部可信性自省机制提供了描述基础，并为

实体自明性的实现建立了良好的途径。建立了网构软件实体可信性自主演化机制，既可保障本实体自身可信性，又能对实体间信任关系及协同行为予以支持。强可信智能实体的建立，形成了最初的信任基，为实体间信任链的建立奠定了良好的基础。

（2）在网构软件实体协同层面，对目前的网构软件信任及演化模型进行了研究与分析，通过引入契约式设计思想，采用“承诺—评估”机制建立了基于契约的网构软件实体交互模型，对信任契约中的前置条件、后置条件、不变式进行了定义，保证了交互双方的可信关联关系。在此基础上提出了基于评估的信任衰减过程，通过建立贝叶斯网络来综合外部环境、实体自身条件情况对主观信任值的影响，使信任传递过程中信任衰减参数的计算更加客观、准确。这些研究成果为实体间的信任链提供了信任传递过程中的可信认证机制，使得多个实体之间形成了逐级信任认证的关系。通过实体模型与实体间信任关系模型的综合实验表明，强可信智能实体模型能够准确感知内部及外部环境的变化并做出自主响应，实体间的信任约束机制也可对实体的交互进行良好保证。

（3）在网构软件系统层面，在分析网构软件体系结构的基础上，建立了系统结构模型，并根据此模型提出了基于分层 Petri 网的网构软件可信性演化模型。研究了 Petri 网的基本概念和性质，并结合网构软件的实际情况对 Petri 网中元素的概念进行了进一步限定，以对系统整体组成结构与各实体内部契约协商策略两方面进行描述。通过上层 Petri 网描述了系统整体组成结构，反映了实体间信任关系的演化；通过下层 Petri 网描述了各实体内部契约协商策略，反映了实体内部机制的运行与变化。通过实例对所提模型进行了验证。该模型对网构软件实体间的信任协商与演化过程进行了持续的管理，并对网构软件演化过程中的可

信性演化情况做出了统一的描述，为网构软件及其实体的设计、开发及部署提供了技术保障与支持。

（4）在网构软件系统测试与可信性评估层面，对贝叶斯网络进行了研究，根据前文建立的系统结构模型，提出了基于贝叶斯网络的网构软件可信性评估体系，并给出了静态指标、动态指标及系统整体可信性的计算方法。该评估体系涵盖了可信性与其特征属性之间的关系，以及网构软件的整体系统与其各组成实体之间的关系。从这两方面关系入手，使用贝叶斯网络对网构软件实体可信性的各特征属性进行综合评估，并采用自底向上的方法，逐层计算，最终得到系统整体的可信性。通过实际运行系统的实验表明，该模型可以正确、有效地评估网构软件系统的可信性，为用户进行系统评估及选择第三方实体的最优组合方案提供了良好的依据。

可信性保证方法与技术是网构软件的一个重要研究领域，目前的研究还不够成熟。未来的研究还需要从以下方面进一步深入。

（1）网构软件实体间信任衰减参数计算方法中的条件概率表使用的数据，取值均为二值，非是即否，系统可信性评估模型中也存在这样的情况。今后可对软件质量评估相关规范、标准进行研究，并能够使用[0，1]区间的数值对条件概率值进行量化，将有望使计算结果更加精确和客观。

（2）由于实验环境所限，模型的性能测试还不够充分，需要建立可信智能实体协同及演化的支撑平台，以支持文中提出的实体交互、演化、协同的机制。并在统一的平台环境下，从实际运行使用的角度出发，对系统完成大规模、长时间、多样性作业的能力进行全面细致的测试，同时进一步完善可信性评估模型，使其更符合实际情况。

（3）对于实体可信性情况及实体间信任契约描述方面的研究还不够深入，形式化的描述主要限于语法层面。今后还需要进一步细化形式化描述中各元素的定义，以便能够从语义层面深入描述与分析网构软件实体个体及各实体间交互的可信性情况，为建立可信智能实体协同及演化的支撑平台提供支持。

参 考 文 献

[1] 吕建, 马晓星, 陶先平, 等. 网构软件的研究与进展. 中国科学：信息科学, 2006, 36(10): 1037-1080.

[2] Dahl O J, Nygaad K. SIMULA—an Algol-based simulation language. Commun ACM, 1966, 9: 671-678.

[3] Booch G. Object-Oriented Analysis and Design with Applications. Reading: Addison-Wesley, 1994.

[4] Bachman F, Bass L, Buhman C. Technical Concepts of Component-Based Software Engineering. CMU/SEI-2000-TR-008, ESC-TR-2000-007, USA 2000.

[5] OMG. CORBA Components 3.0. Object Management Group. 2002.

[6] Szyperski C, Gruntz D, Murer S. Component-Software: Beyond Object-Oriented Programming. Pearson Education Limited. Boston: Addison-Wesley/ACM Press, 1997.

[7] Alonso G, Casati F, Kuno H, et al. Web Services: Concepts, Architectures and Applications. Heideberg: Springer, 2004.

[8] Cerami E. Web Services Essentials. Sebastopol, Calif: O'Reilly. 2002.

[9] Huhns M, Singh M P. Service-oriented computing: Key concepts and principles. IEEE Internet Comput, 2005, 9(1): 75-81.

[10] Khalaf R, Mukhi N, Weerawarana S. Service-oriented composition in BPEL4WS. In: Proceedings of the 2003 World Wide Web conference.

Palm Springs: IEEE Computer Society, 2003.

[11] Stal M. Using architectural patterns and blueprints for service-oriented architecture. IEEE Software, 2006, 23(2): 54-61.

[12] OMG. Unified modeling language(UML), version 1.3. Object Management Group, 1999.

[13] Bachman F, Bass L, Buhman C, et al. Technical Concepts of Component-Based Software Engineering, CMU. Technical Report CMU/SEI-2000.2000.

[14] Szyperski C, Gruntz D, Murer S. Component-Software: Beyond Object-Oriented Programming. 2nd ed. Pearson Education Limited. Boston: Addison-Wesley/ACM Press, 2002.

[15] Wooldridge M, Jennings N. Intelligent agents: Theory and practice. Knowl Eng Rev, 1995, 10(2): 115-152.

[16] Weiss G. Multiagent Systems: A Modern Approach to Distributed. Massachusettes: MIT Press, 1999.

[17] 杨芙清, 吕建, 梅宏. 网构软件技术体系: 一种以体系结构为中心的途径. 中国科学: 信息科学, 2008, 38(6): 818-828.

[18] 杨芙清, 梅宏, 吕建, 等. 浅论软件技术发展. 电子学报, 2002, 30(12A): 1901-1906.

[19] Milner R. Theories of the global ubiquitous computer. In: Foundations of Software Science and Computation Structures, LNCS 2987. Berlin: Springer-Verlag, 2004.

[20] Shaw M. Sufficient correctness and homeostasis in open resource coalitions: How much can you trust your software system. In: Proce-

eding of the Fourth International Software Architecture Workshop, 2000.

[21] 赵丽娜, 张引, 叶修梓, 等. 基于 PZP 网络的网构软件自适应性研究. 浙江大学学报(工学版), 2005, 42(8):1316-1322.

[22] 孙熙, 庄磊, 刘文, 等. 一种可定制的自主构件运行支撑框架. 软件学报, 2008, 19(6): 1340-1349.

[23] 刘文, 孙熙, 焦文品, 等. 一种基于自主构件的网构软件协作框架. 计算机研究与发展, 2006, 46(Suppl.):217-221.

[24] 吕建, 陶先平, 马晓星, 等. 基于 Agent 的网构软件模型研究. 中国科学 E 辑: 信息科学, 2005, 35(12): 1233-1253.

[25] 常志明, 毛新军, 齐治昌. 基于 Agent 的网构软件构件模型及其实现. 软件学报, 2008, 19(5): 1113-1124.

[26] 丁博, 王怀民, 史殿习, 等. 一种支持软件可信演化的构件模型. 软件学报, 2011, 22(1): 17-27.

[27] Papadopoulos G A, Arbab F. Coordination models and languages. Centrum voor Wiskunde en Information(CWI). Amsterdam: Software Engineering(SEN), 1998.

[28] 侯丽珊, 金芝. 基于环境变迁的构件组合模型. 电子学报, 2005, 33(12A): 2370-2375.

[29] Shaw M, Clements P. The golden age of software architecture. IEEE Software, 2006, 23(2): 31-39.

[30] Kumar N, Misurda J, Childers B R, et al. Instrumentation in software dynamic translators for self-managed systems. In: Proceedings of the 1st ACM SIGSOFT Workshop on Self-managed Systems. New York:

SCM Press, 2004, 90-94.

[31] 马晓星, 余萍, 陶先平, 等. 一种面向服务的动态协同架构机器支撑平台. 计算机学报, 2005, (4): 467-477.

[32] Tao X, Lu J, Ma X. ARTeMIS: a multi-model coordination middleware based on mobile agent. Technical Report. State Key Laboratory for Novel Software Technology, Nanjing: Nanjing University, 2003.

[33] Blaze M, Feigenbaum J, Lacy J. Decentralized trust management. In: Proc 17th Symposium on Security and privacy. Los Alamitos: IEEE Computer Society Press, 1996, 164-173.

[34] Zhang L, Xu F, Wang Y, et al. Design and implementation of cascaded monitor in trust management system. Nanjing University, 2006.

[35] Grandison T, Sloman M. Trust management tools for internet application. In: Proceeding of the First International Conference on Trust Management. Berlin: Springer-Verlag, 2002.

[36] Tesauro G, Chess D M, Walsh W E, et al. A multi-agent systems approach to autonomic computing. In: Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent System-Volume 1. New York: IEEE Computer Society, 2004, 467-471.

[37] Appavoo J, Hui K, Soules C A N, et al. Enabling autonomic behavior in systems software with hot swapping. IBM Syst J, 2003(42): 60-76.

[38] 张焕国, 罗捷, 金刚, 等. 可信计算研究进展.武汉大学学报(理学版), 2006(52): 513-518.

[39] Trusted Computing Group(TCG). TCPA Main Specification, Version 1.1b. 2002.

[40] Anderson J P. Computer Security Technology Planning Study. ESD-TR-73-51, Vol. I, AD-758 206, ESD/AFSC, Hanscom AFB, Bedford MA, October 1972.

[41] Department of Defense Computer Security Center. Department Of Defense Trusted Computer System Evaluation Criteria. DoD 5200.28-STD. USA: DOD, 1985.

[42] National Computer Security Center. Trusted Network Interpretation Of The Trusted Computer System Evaluation Criteria. NCSC-TG-005. USA:DOD, 1987.

[43] National Computer Security Center. Trusted Database Management System Interpretation of the TCSEC. NCSC-TG-021, 1991.

[44] Trusted Computing Group. TCG. https://www.trustedcomputinggroup.org.

[45] Trusted Computing Group. TCG 规范列表. https://www.trustedcomputinggroup.org/specs/.

[46] European Multilaterally Secure Computing Base(EMSCB). Towards trustworth systems with open standards and trusted computing. http://www.opentc.org/.

[47] 张焕国, 刘服珍, 余发江, 等. 一种新型嵌入式安全模块. 第一届中国可信计算与信息安全学术会议论文集. 武汉大学学报(理学版), 2004, 50(s1): 7-11.

[48] 国家密码管理局. 可信计算密码支撑平台功能与接口规范. 2007.

[49] 刘克, 单志广, 王戟, 等. “可信软件基础研究”重大研究计划综述. 中国科学基金—学科进展与展望, 2008, 3: 145-151.

[50] Trusted Computing Group. TCG Generic Server Specification. Version 1.0. TCG, 2005.

[51] Sadeghi A R, Selhorst M, Stueble C, et al. TCG inside? – a note on TPM specification compliance. In: Proceedings of the 1st ACM Workshop on Scalable Trusted Computing. New York: ACM, 2006, 47-56.

[52] 陈火旺, 王戟, 董威. 高可信软件工程技术. 电子学报, 2004, 31: 1934-1938.

[53] TCG Specification Trusted Network Connect – TNC Architecture for Interoperability Revision 1.1. http://www.trustedcomputinggroup.org.

[54] Kuhn U, Selhorst M, Stuble C. Realizing property-based attestation and sealing with commonly available hard- and software. In: Proceedings of the 1st ACM Workshop on Scalable Trusted Somputing (STC'07). New York, NY, 2007, 50-57.

[55] 沈昌祥, 张焕国, 王怀民, 等. 可信计算的研究与发展. 中国科学: 信息科学, 2010, 40(2): 139-166.

[56] ISO/IEC. Information technology-security Techniques-Evaluation Criteria for IT Security. Part 1: Introduction and General Model. 2nd ed. 2005. http://standards.iso.org.

[57] Trusted Computing Group. TCG Architecture Overview, v1.2, 28 April 2004. https: //www.trustedcomputinggroup.org.

[58] Gates B. Trustworthy Computing. Wired News, Jan. 17, 2002.

[59] 林闯. 可信网络研究. 计算机学报, 2005, 28(5): 751-758.

[60] Algirdas A, Jean-Claude L, Brian R, et al. Basic concepts and

taxonomy of dependable and secure computting. IEEE Trans Dependable Secure, 2004, 1(1): 11-33.

[61] 王怀民, 唐扬斌, 尹刚, 等. 互联网软件的可信机理. 中国科学 E 辑:信息科学, 2006, 36(10): 1156-1169.

[62] 邓晓衡, 卢锡城, 王怀民. iVCE 中基于可信评价的资源调度研究. 计算机学报, 2007, 30(10): 1750-1762.

[63] Dong W, Wang J, Zhao C Z, et al. Automating software FMEA via formal analysis of dependence relations. In: Proceedings of the 32nd Annual IEEE International Computer Software and Applications Conference(COMPSAC). New York: IEEE Computer Society, 2008, 490-491.

[64] Lyu M R. Handbook of Software Reliability Engineering. NewYork: IEEE Computer Society Press, McGraw-Hill Book Company, 1996.

[65] Rolland J F, Bodeveix J P, Filali M, et al. AADL modes for space software, data systems. In: Aerospace(DASIA 2008). Palma de Majorca, 2008, 27-30.

[66] Mens T, Demeyer S. Software Evolution. Berlin/Heidelberg: Springer-Verlag, 2008.

[67] Ghoshal S, Manimaran S, Rosu G, et al. Monitoring IVHM systems using a monitor-oriented programming framework. In: Proceedings of the 6th NASA Langley Fomal Methods Workshop(LFM 2008), 2008.

[68] Nahmsuk O. Software Implemented Hardware Fault Tolerance. California: Stanford University, 2001.

[69] 梅宏, 王千祥, 张路, 等. 软件分析技术进展. 计算机学报, 2009(9): 1697-1710.

[70] Clarke E M, Grumberg O, Peled D A. Model Checking. Massachusetts: MIT Press, 2000.

[71] Boldyreff C, Nutter D, Rank S, et al. Environments to support collaborative software engineering. In: Proceedings of the 2nd Workshop on Cooperative Supports for Distributed Software Engineering Processes, 2003, 25-28.

[72] Oreizy P, Medvidovic N, Taylor R N. Runtime software adaptation: framework, approaches, and styles. ICSE, 2008, 899-910.

[73] Ruhe M G, Eberlein A. COTS selection: past, present, and future. In: Proceedings of the 14th Annual IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, ECBS ’07. Washington: IEEE Computer Society, 2007, 103-114.

[74] Feiler N L, Gabriel P, Goodenough R, et al. Ultra-Large-Scale Systems: the Software Challenge of the Future. Software Engineering Institute. Pittsburgh, PA: Carnegie Mellon University, 2006.

[75] 王远, 吕建, 徐锋, 等. 一个适用于网构软件的信任度量及演化模型. 软件学报, 2006, 17(4): 682-690.

[76] Blaze M, Feigenbaum J, Ioannidis J, et al. The KeyNote Trust Management System Version 2, Internet RFC 2704. 1999.

[77] Chu Y H, Feigenbaum J, LaMacchia B, et al. REFEREE: Trust management for Web applications. World Wide Web J, 1997, 2(2): 127-139.

[78] Li N, Winsborough W H, Mitchell J C. Distributed credential chain discovery in trust management. J Comput Secur, 2003, 11(1): 35-86.

[79] Freudenthal E, Pesin T, Port L. dRBAC: Distributed role-based access control for dynamic coalition environments. In: Proc 22nd International Conference on Distributed Computing Systems(ICDCS'02). Vienna: IEEE, 2002, 294-306.

[80] English C, Wagealla W, Nixon P, et al. Trusting collaboration in global computing systems. Lecture Notes in Computer Science 2003, 123-149.

[81] Kamvar S, Schlosser M, Garcia-Molina H. The eigentrust algorithm for reputation management in P2P networks. In: Proceedings of the 12th International Conference on World Wide Web 2003, 640-651.

[82] 张林, 徐锋, 王远, 等. 一种信任管理系统中层次式 monitor 机制的设计与实现. 南京大学学报(自然科学), 2007, 43(2): 191-198.

[83] 李佳伦, 谷利泽, 杨义先. 一种具有时间衰减和主观预期的P2P网络信任管理模型. 电子与信息学报, 2009, 31(11): 2786-2790.

[84] 罗鑫, 杨义先, 胡正名, 等. 开放网络环境中的信任管理框架. 北京邮电大学学报, 2009, 32(1): 126-130.

[85] Abdul-Rahman A, Hailes S. Using recommendations for managing trust in distributed systems. In: IEEE Malaysia International Conference on Communication Citeseer 1997.

[86] 胡建理, 吴泉源, 周斌, 等. 一种基于反馈可信度的分布式 p2p 信任模型. 软件学报, 2009, 20(10): 2885-2898.

[87] Abdul-Rahman A, Hailes S. A distributed trust model. In: Proceeding

of the 1997 New Security Paradigms Workshop. Cumbria: ACM Press, 1998, 48-60.

[88] Beth T, Borcherding M, Klein B. Valuation of trust in open NetWork. In: Proc. European Symposium on Research in Security(ESORICS). Brighton: Springer-Verlag, 1994, 3-18.

[89] Jøsang A. A model for trust in security systems. In: Proceedings of the Second Nordic Workshop on Secure Computer Systems, 1997.

[90] 徐锋, 吕建, 郑玮, 等. 一个软件服务协同中信任评估模型的设计. 软件学报, 2003, 14(6): 1043-1051.

[91] Houser D, Wooders J. Reputation in auctions: theory and evidence from eBay. University of Arizona Working Paper, University of Arizona, 2000.

[92] Sepandar DK, Mario TS, Hector GM. The EigenTrust algorithm for reputation management in P2P networks. In: Proc. of the 12th Int'l Conf. on World Wide Web. Budapest: ACM Press, 2003, 640-651.

[93] Zhou RF, Hwang K. PowerTrust: A Robust and Scalable Reputation System for Trusted Peer-to-Peer Computing. IEEE Transactions on parallel and distributed systems, 2007, 18(4): 460-473.

[94] Xiong L, Liu L. PeerTrust: Supporting Reputation-Based Trust for Peer-to-Peer Electronic Communities. IEEE Transaction on knowledge data engineering, 2004, 16(7): 843-857.

[95] 田春岐, 邹仕洪, 田慧蓉, 等. 一种基于信誉和风险评价的分布式 P2P 信任模型. 电子与信息学报, 2007, 29(7): 1628-1632.

[96] Jøsang A, Haller J. Dirichlet Reputation Systems. In: Proceedings of

the 2nd International Conference on Availability, Reliability and Security Vienna. Los Vaqueros: IEEE computer society, 2007, 112-119.

[97] Meyer B, Zürich E T H, Barbara S. The grand challenge of Trusted Components. In: Proceedings of the 25th International Conference on Software Engineering, 2003.

[98] ISO/IEC 9126 Directives, ISO/IEC 9126-1:2001 Software engineering -- Product quality-- Part 1: Quality model, ISO/IEC 9126, ISO, 2001.

[99] Yacoub S M, Ammar H H. A methodology for architecture-level reliability risk analysis. IEEE Transactions on Software engineering, 2002, 28(6): 529-547.

[100] Hamlet D, Mason D, Woit D. Theory of Software Reliability Based on Components. In: Proceedings of the 23rd International Conference on Software Engineering(ICSE'01), 2001.

[101] Roshandel R, Medvidovic N. Multi-View Software Component Modeling for Dependability. Lecture Notes in Computer Science Volume 3069/2004.

[102] Reussner R H, Schmidt H W, Poernomo I H. Reliability prediction for component-based software architectures. Journal of systems and software, 2003, 66: 241-252.

[103] Guerra P A C, Filho F C, Pagano V A, Rubira C M F. Structuring Exception Handling for Dependable Component-Based Software Systems. In: Proceedings of the 30th EUROMICRO Conference(EUROMICRO'04), 2004.

[104] Bobbio A, Portinale L, Minichino M. Improving the analysis of dependable systems by mapping fault trees into Bayesian networks. Reliability Engineering and System Safety, 2001,71(3): 249-260.

[105] 吴国全, 魏峻, 黄涛. 基于非确定性推理的网构软件服务质量动态评估方法. 软件学报, 2008, 19(5): 1173-1185.

[106] 郭树行, 兰雨晴, 金茂忠. 软件构件的可信保证研究. 计算机科学, 2007, 34(5): 243-246.

[107] 毛晓光, 邓勇进. 基于构件软件的可靠性通用模型. 软件学报, 2004, 15(1): 27-32.

[108] 陈锦富. 基于错误注入的构件安全性测试理论与技术研究. 华中科技大学, 博士论文, 2009.

[109] 陈锦富, 卢炎生, 谢晓东. 一种构件安全测试错误注入模型. 计算机研究与发展, 2009, 46(7): 1127-1135.

[110] Wang Y, Vassileva J. A review on trust and reputation for Web service selection. In: Proc. of the 27th Int'l Conf. on Distributed Computing Systems Workshops, 2007.

[111] Vu L H, Hauswirth M, Aberer K. Qos-Based service selection and ranking with trust and reputation management. In: Proc. of the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE. LNCS 3760, 2005, 466-483.

[112] Diamadopoulou V, Makris C, Panagis Y, et al. Techniques to support web service selection and consumption with QoS characteristics. Journal of Network and Computer Applications, 2008, 31(2): 108-130.

[113] Zhang J, Xu D. A mobile agent-supported Web services testing platform. In: Proceedings of the IEEE/IFIP International Conference on Embedded and Ubiquitous Computing. Washington, DC, USA: IEEE Computer Society, 2008, 637-644.

[114] Shao L S, Zhao J F, Xie T, et al. User-perceived service availability: a metric and an estimation approach. In: Proceedings of the IEEE International Conference on Web Services, Los Angeles, 2009. Washington, DC, USA: IEEE Computer Society, 2009: 647-654.

[115] Bai X Y, Dong W L, Tsai W T, et al. WSDL-based automatic test case generation for Web services testing. In: Proceedings of the IEEE International Workshop on Service-Oriented System Engineering, Los Alamitos, 2005. Washington, DC, USA: IEEE Computer Society, 2005, 215-220.

[116] Maximilien E M, Singh M P. A framework and ontology for dynamic Web services selection. IEEE Internet Computing, 2004,8 (5): 84-93.

[117] 朱曼玲, 金芝. 一种服务 Agent 的可信性评估方法. 软件学报, 2011, 22(11): 2593-2609.

[118] 孟琳琳, 赵伟男, 刘旭东, 等. Web 服务可信证据收集与评估机制研究. 计算机科学与探索, 2011, 5(7), 642-651.

[119] Guannan Si, Jufeng Yang, Jing Xu, et al. An Evaluation Model for Dependability of Internet-scale Software on Basis of Bayesian Networks, In: the IEEE Computer Software and Applications Conference(COMPSAC 2012), 2012.

[120] 司冠南, 任宇涵, 许静, 等. 基于贝叶斯网络的网构软件可信性评估模型. 计算机研究与发展, 2012, 49(5).

[121] 吕建, 张鸣, 廖宇, 等. 基于移动 Agent 技术的构件软件框架研究. 软件学报, 2000, 11(8): 1018-1024.

[122] 胡海洋, 杨玫, 陶先平, 等. Cogent后组装机制的研究实现. 电子学报, 2002, 30(12): 1823-1827.

[123] Lu J. Some research on componentware frameworks based on mobile agent technology. ACM SIGSOFT Software Engineering Notes, 2004, 29(2): 8-15.

[124] Lu J, Li Y J, Ma X X, et al. A hierarchical framework for parallel seismic applications. Communications of ACM, 2000, 43(10): 55-59.

[125] Ma X X, Lu J, Tao X P, et al. A mobile-agent-based approach to software coordination in the HOOPE system. Science in China, Series F, 2002, 45(3): 203-219.

[126] 吕建, 陶先平, 马晓星, 等. 基于 Agent 的多模式协同中间件 ARTEMIS-M3C. 南京大学计算机软件新技术国家重点实验室技术报告, 2004.

[127] Mei H, Huang G. PKUAS: An architecture-based refective component operating platform. In: Proceedings of the 10th IEEE Int'l Workshop on Futre Trends of Distributed Computing Systems (FTDCS). Los Alamitos: IEEE Computer Society, 2004, 163-169.

[128] Pan Y, Wang L, Zhang L, et al. Relevancy based sematic interoperation of reuse repositories. In: Proceedings of the 12th

ACM SIGSOFT Symp. On Foundations of Software Engineering (FSE-12). New York: ACM Press, 2004, 211-220.

[129] 梅宏, 黄罡, 赵海燕, 等. 一种以软件体系结构为中心的网构软件开发方法. 中国科学 E 辑: 信息科学, 2006, 36(10): 1100-1126.

[130] Vestal S. A cursory overview and comparison of four architecture description languages. Technical Report, Honeywell Technology Center, 1993.

[131] Clements P C. A survey of architecture description language. In: Proceeding of the 8th International Workshop on Software Specification and Design, 1996.

[132] Medvidovic N, Taylor R N. A classification and comparison framework for software architecture description language. IEEE Transactions On Software Engineering, 2002, 26(1): 70-93.

[133] 朱雪阳, 唐稚松. 基于时序逻辑的软件体系结构描述语言 XYZ/ADL. 软件学报, 2003, 14(4): 714-720.

[134] Mei H, Chen F, Wang Q X, et al. ABC/ADL: An ADL Supporting Component Composition. LNCS 2495, Springer-Verlag, 2002, 38-47.

[135] Garlan D, Monroe R T. Acme: Architectural Description of Component-Based System. In: Proceedings of CASCON '97, 1997.

[136] World Wide Web Consortium. REC-xml-20081126. Extensible markup language(XML)1.0(fifth edition). Availiable at: http://www.w3.org/TR/2008/REC-xml-20081126/, 2008.

[137] World Wide Web Consortium. REC-xmlschema-0-20041028.

Availiable at: http://www.w3.org/TR/ REC-xmlschema-0-20041028/, 2004.

[138] 张世琨, 张文娟, 常欣, 等. 基于软件体系结构的可复用构件制作和组装. 软件学报, 2001, 12(9): 1351-1359.

[139] Avgustinov P, Tibble J, Bodden E, et al. Aspect for Trace Monitoring.Formal Approaches to Testing Systems and Runtime Verification, Seattle, WA, USA, 2006, 20-39.

[140] 文静, 王怀民, 应时, 等. 支持运行监控的可信软件体系结构设计方法. 计算机学报, 2010, 33(12): 2321-2334.

[141] Mui L. Computational models of trust and reputation: agents, evolutionary games, and social Networks [Ph D dissertation]. Combridge: Massachusetts Institute of Technology, 2003.

[142] 王远, 吕建, 徐锋, 等. 一种面向网构软件体系结构的信任驱动服务选取机制. 软件学报, 2008, 19(6): 1350-1362.

[143] Walket D M. The Oxford Companion to Law. Oxford University Press, 1980.

[144] Meyer B. Applying “design by contract”. Computer, 1992, 25(10): 40-51.

[145] Meyer B. Object Oriented Software Construction. Prentice-Hall, 1998.

[146] Lackner M, Krall A, Puntigam Franz. Supporting Design by Contract in Java. Journal of Object Technology, 2002, 1(3).

[147] Beugnard A, Jezequel J M, Plouzeau N. Make components contract aware. Computer, 1999, 32(7): 38-45.

[148] Ling S, Poernomo I, Schmidt H. Describing web service architectures through design-by-contract. ISCIS 2003, 2003, 1008-1018.

[149] Luders F, Lau K K, Ho S M. Specification of software components, building reliable component-based software systems. Boston, London: Artech House Computing Library, 2002.

[150] Murata T. Petri nets: properties, analysis and application. Proceedings of the IEEE, 1989, 77(4), 541-580.

[151] 吴哲辉. Petri 网导论. 北京：机械工业出版社, 2006.

[152] Barkaoui K, Pradat-peyre J. Verification in comcurrent programming with Petri nets structural techniques. In: Proceedings of the 3rd Inter IEEE High-Assurance System Engineering Symposium. Los Alamitos, CA: IEEE Computer Society Press, 1998, 124-133.

[153] Yang Y P, Tan Q P, Xiao Y. Verifying Web Services Composition Based on Hierarchical Colored Petri Nets. In: Proceedings of the first international workshop on Interoperability of heterogeneous information systems. ACM, 2005, 47-54.

[154] van der Aalst W M P. The application of Petri nets to workflow management. The Journal of Circuits t Systems and Computers, 1998, 8(1): 21-66.

[155] Zhou M C, Leu M C. Modeling and performance analysis of a flexible PCB assembly station using Petri nets. Trans ASME J Electron Packag, 1991, 113(4): 410-416.

[156] 韩耀军, 蒋昌俊, 罗雪梅. 基于 Petri 网合成与化简的分布式数据库系统并发控制的死锁检测. 小型微型计算机系统, 2004,

25(5): 821-826.

[157] 陆文, 徐锋, 吕建. 一种开放环境下的软件可靠性评估方法. 计算机学报, 2010, 33(3): 452-462.

[158] Winsborough W H, Seamons K E, Jones V E. Automated trust negotiation. In:DARPA Information Survivability Conf. and Exposition. New York: IEEE Press, 2000, 88-102.

[159] Walker D D, Mercer E G, Seamons K E. Or Best Offer: A Privacy Policy Negotiation Protocol. In: Proc. of the 9th Int'l Workshop on Policies for Distributed Systems and Networks. New York: IEEE Computer Society Press, 2008, 173-180.

[160] Li J T, Li N H, Winsborough W H. Automated trust negoti-ation using cryptographic credentials. In: ACM Transac-tions on Information and System Security, 2009(13), 46-57.

[161] 李建欣, 怀进鹏. COTN:基于契约的信任协商系统. 计算机学报, 2006(8), 1290-1300.

[162] 张妍, 冯登国. 吝啬语义信任协商. 计算机学报, 2009(10), 1989-2003.

[163] He Y, Zhu M L. A complete and efficient strategy based on petri net in automated trust negotiation. In: Proceedings of the 2nd international conference on Scalable information systems, Brussels: ACM, 2007, 75-81.

[164] Johnson W, Mudumbai S, Thompson M. Authorization and attribute certificates for widely distributed access control. In: IEEE Proc. of the 7th Workshop on Enabling Technologies: Infrastructure for

Collaborative Enterprises. Washington: IEEE Computer Society Press, 1998, 340-345.

[165] Seamons K E, Winslett M, Yu T, et al. Requirements for policy languages for trust negotiation. In: Proc. of the 3rd Int'l Workshop on Policies for Distributed Systems and Networks(POLICY 2002). Washington: IEEE Computer Society Press,2002, 68-79.

[166] Yu T, Winslett M. A unified scheme for resource protection in automated trust negotiation. In: Proc. of the 2003 IEEE Symp. on Security and Privacy. Washington: IEEE Computer Society Press, 2003, 110-122.

[167] Laprie J C, ed. Dependability: Basic Concepts and Terminology. Springer-Verlag, 1992.

[168] 金茂忠, 高仲仪, 刘超, 等. 软件构件产品质量 第一部分：质量模型. 中华人民共和国电子行业标准 SJ/T 11374—2007. 中华人民共和国信息产业部, 2007.

[169] Nilsson N. 人工智能. 郑扣根, 庄越挺, 译. 北京：机械工业出版社, 2000.

[170] Stephenson T A. An introduction to Bayesian Network theory and usage. IDIAP-RR 00-03, 2000.

[171] Kim J H, Pearl J. A computation model for causal and diagnostic reasoning in inference systems. In: Proeeedings of the 8th International Joint Conference on AI, Los Angeles, 1983, 190-193.

[172] Charniak E. Bayesian Networks without tears. AI Magazine, 1991, 12(4): 50-63.

[173] 黎锁平. 运用蒙特卡罗方法求解随机性问题. 甘肃工业大学学报, 2001, 27(2): 95-97.

[174] Jensen F V. An introduction to Bayesian Networks. UCL Press Ltd., London,1996.

[175] 吴欣, 郭创新. 基于贝叶斯网络的电力系统故障诊断方法. 电力系统及其自动化学报, 2005, 17(4): 11-15.

[176] 史建国, 高晓光, 李相民. 基于离散模糊动态贝叶斯网络的空战态势评估及仿真. 系统仿真学报, 2006, 18(5): 1093-1096.

[177] Heckerman D, Mamdani A, Wellman M. Real-world applications of Bayesian networks. Communications of the ACM, 1995, 38(3): 38-45.

[178] Buede D M, Tatman J A, Bresnick T A. Introduction to Bayesian Networks. http://www.ecse.rpi.edu/Homepages/qji/TutFinbd.ppt.

[179] Getoor L, Friedman N, Koller D, et al. Learning probabilistic relational models with structural uncertainty. In Proceedings of the International Conference on Machine Learning. Morgan Kaufman, 2001, 170-177.

[180] Getoor L, Segal E, Taskar B, et al. Probabilistic models of text and link structure for hypertext classification. In IJCAI Workshop on Text Learning: Beyond Supervision, 2001.

[181] Torres-Toledano J G, Sucar L E. Bayesian Networks for Reliability Analysis of Complex Systems. Proceedings of the 6th Ibero-American Conference on AI: Progress in Artificial Intelligence, vol 1484 of

LNCS, 1998, 195-206.

[182] Arroyo G, Sucar L, Villavicencio A. Probabilistic temporal reasoning and its application to fossil power plant operation. Expert Systems with Applications, 1998, 15, 317-324.

[183] Bai C G. Bayesian network based software reliability prediction with an operational profile. Journal of Systems and Software, 2005, 77(2): 103-112.

[184] Axel B, Helminen A. A Bayesian belief network for reliability assessment. SAFECOMP 2001, vol 2187 of LNCS, 2001, 35-45.

[185] Boudali H, Dugan J B. A discrete-time Bayesian network reliability modeling and analysis framework. Reliability Engineering and System Safety, 2005, 87(3): 337-349.

[186] Bouissou M, Martin F, Ourghanlian A. Assessment of a Safety Critical System Including Software: a Bayesian Belief Network for Evidence Sources. Proceeding of the Reliability and Maintainability Symposium(RAMS'99), 1999, 142-150.

[187] Montani S, Portinale L, Bobbio A, et al. Codetta-Raiteri. A tool for automatically translating Dynamic Fault Trees into Dynamic Bayesian Networks. Proceeding of the Reliability and Maintainability Symposium(RAMS 2006), 2006, 434-441.

[188] Weber P, Munteanu P, Jouffe L. Dynamic Bayesian Networks modelling the dependability of systems with degradations and exogenous constraints. Proceeding of the 11th IFAC Symposium on Information Control Problems in Manufacturing(INCOM'04), 2004.

[189] Boudali H, Dugan J B. A continuous-time Bayesian network reliability modeling and analysis framework. IEEE Transaction on Reliability, 2006, 55(1): 86-97.

[190] Voas J. Fault injection for the Masses. IEEE Computer, 1997, 30(12): 129-130.

[191] Hsueh M C, Tsai T K, Lyer R K. Fault injection techniques and tools. IEEE Computer, 1997, 30(4): 75-82.

[192] Delamam M E, Maidonado J C. Mathur A P. Interface Mutation: an approach for integration testing. IEEE Transactions on Software Engineenng, 2001, 27(3): 228-247.

[193] Avres D, Arlat J, Laprie J C, et al. Fault injection for formal testing of fault tolerance. IEEE Transactions On Reliability, 1996, 45(3): 443-455.

[194] Barbosa R, Silva N, Duraes J, et al. Verification and validation of(Real Time)COTS products using fault injection techniques. In: Proceedings of Sixth Intemational IEEE Conference on Commercial-off-the-Shelf(COTS)-Based Software Systems(ICCBSS'07). Canada: IEEE CS, 2007, 233-242.

[195] Bieman J M, Dreilinger D, Lijun L. Using fault injection to increase software teste overage. In: Proceedings of the Seventh International Symposium on Software Reliability Engineenng(ISSRE'96). IEEE Computer Society, 1996, 166-174.

[196] Regina L O M, Eliane M, Naaliel Vicente M. Fault injection approach based on dependence analysis. In: Proeeedings of the 29th Annual

International Computer Software and Applications Conference(COMPSAC'05). IEEE Computer Society, Scotland, 2005, 181-188.

[197] Yao W H, Shih K H, Tsung P L, et al. Web application security assessment by fault injection and behavior monitoring. In: Proceedings of the 12th International conference on World Wide Web(WWW2003). ACM: Budapest, Hungary, 2003, 148-159.

[198] 宋波.Web 应用交互的建模和测试用例生成[博士学位论文]. 上海：上海大学，2010.

[199] 杜经农. 基于 Web 的应用软件安全漏洞测试方法研究[博士学位论文]. 武汉：华中科技大学，2010.

[200] The Open Web Application Security Project. OWASP Top 10-2010：the ten most critical web application security risks. https://www.owasp.org/index.php/Category:OWASP_Top_ Ten_Project. 2010.

[201] William G.J. Halfond, Shauvik Roy Choudhary, Alessandro Orso. Improving Penetration Testing Through Static and Dynamic Analysis. In: Proceedings of the Second IEEE International Conference on Software Testing, Verification and Validation. West Sussex, UK: John Wiley and Sons Ltd, 2011, 195-214.

[202] Jason Bau, Elie Bursztein, Divij Gupta, et al. State of the Art: Automated Black-Box Web Application Vulnerability Testing. In: Proceedings of the 2010 IEEE Symposium on Security and Privacy. Piscataway, USA: IEEE, 2010, 332-345.

[203] Adam Doup´e, Marco Cova, Giovanni Vigna. Why Johnny Can't Pentest: An Analysis of Black-box Web Vulnerability Scanners. In:

Proceedings of the 7th GI International Conference on Detection of Intrusions and Malware and Vulnerability Assessment. Bonn, Germany: Springer Verlag, 2010, 111-131.

[204] 沈寿忠. 基于网络爬虫的 SQL 注入与 XSS 漏洞挖掘[硕士学位论文]. 西安：西安电子科技大学，2009.

[205] 曾凡平. 软件漏洞测试若干问题研究[博士学位论文]. 长沙：国防科技大学，2009.

[206] William G.J. Halfond, Alessandro Orso, Panagiotis Manolios. WASP: Protecting web applications using positive tainting and syntax-aware evaluation. IEEE Transactions on Software Engineering, 2008, 34(1):65-81.

[207] 蒋廷耀,王训宇,马凯,关国翔. 基于 EAI 和 AOP 的软件安全测试及应用研究. 计算机科学，2009, 36（4）：169-171.

[208] Jie Wang, Raphael C.-W. Phan, John N. Whitley, et al. Augmented Attack Tree Modeling of SQL Injection Attacks. In: Proceedings of the 2nd IEEE International Conference on Information Management and Engineering. Chengdu, China: IEEE, 2010, 182-186.

[209] Aaron Marback, Hyunsook Do, Ke He, et al. Security Test Generation using Threat Trees. In: Proceedings of the ICSE Workshop on Automation of Software Test (AST '09). Vancouver, BC, Canada: IEEE, 2009, 62-69.

[210] William G.J. Halfond, Viegas J, Alessandro Orso. A classification of SQL-injection attacks and countermeasures. In: Proceedings of the International Symposium on Secure Software Engineering. Washi-

ngton, DC, U.S.A., 2006.

[211] Fraser Gordon,Gargantini Angelo. An Evaluation of Model Checkers for Specification Based Test Case Generation. In: International Conference on Software Testing Verification and Validation,ICST'09. Denver, CO, USA: IEEE, 2009, 41-50.

[212] 奚和平. 基于构件的软件测试模型及方法. 解放军理工大学学报(自然科学版)，2006, 7(3)：632-637.

[213] 路晓丽,董云卫,赵宏斌.一种面向对象的 Web Application 测试模型.计算机科学, 2010, 37(7):134-136.

[214] 钱忠胜，缪淮扣. 基于规格说明的若干逻辑覆盖测试准则. 软件学报,2010, 21(7):1536-1549.

[215] 钱忠胜，缪淮扣，陈圣波．基于 ORD 和 FSM 的 Web 应用的建模与测试．计算机科学，2008, 35(9): 278-281.

[216] 许蕾，陈林，徐宝文. 用户需求驱动的 Web 服务测试. 计算机学报，2011, 34(6): 1029-1040.

[217] 董文莉, 胡建华. 基于 BPEL 的 Web Service 组合的数据流分析测试方法. 软件学报，2009, 20(8): 2102-2112.

[218] 姜瑛，辛国茂，单锦辉, 等. 一种 Web 服务的测试数据自动生成方法. 计算机学报，2005, 28(4): 568-577.

[219] Axel Belinfante, Lars Frantzen, Christian Schallhart. Tools for Test Case Generation. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2005, 3472: 391-483.

[220] Mark Utting, Alexander Pretschner, Bruno Legeard. A taxonomy of

model-based testing. Software Testing Verification and Reliability, 2011.

[221] Andreas L. Opdahl, Guttorm Sindre. Experimental comparison of attack trees and misuse cases for security threat identification. Information and Software Technology, 2009, 51 (5) : 916-932.

[222] David Byers, Nahid Shahmehri. Unified modeling of attacks, vulnerabilities and security activities. In: Proceedings of the 2010 ICSE Workshop on Software Engineering for Secure Systems, Cape Town, South africa: IEEE Press, 2010, 36-42.

[223] Zhengdong Su, Gary Wassermann. The Essence of Command Injection Attacks in Web Applications. ACM SIGPLAN Notices, 2006, 41(1): 372-382.

[224] 蒲石. Web 安全渗透测试研究[硕士学位论文]. 西安：西安电子科技大学，2010.